13th ARRL DIGITAL COMMUNICATIONS CONFERENCE

Bloomington, Minnesota
AUGUST 19-21, 1994

13th ARRL DIGITAL COMMUNICATIONS CONFERENCE

Hosted by: The TwinsLAN ARC

ISBN: 0-87259-483-1

ARRL Order Number: 4831

First Edition

Forward

The American Radio Relay League takes pride in publishing these papers of the 13th ARRL Digital Communications Conference.

This year's papers showcase a broad range of creative ideas. Explorations in HF digital communication continue in both hardware and software. You'll discover several papers concerning G-TOR, the newest HF digital communication mode. There is also a fascinating discussion of a low-cost DSP modem for HF, a preview of HF packet-radio protocol performance and a proposal for amateur emulation of ALE on the HF bands.

The Automatic Packet Reporting System (APRS) also shares the spotlight. APRS is a growing subspecialty within the packet world and its potential is enormous--especially for public service applications.

For those interested in data compression, there are several papers that present interesting alternatives. Two approaches to image compression are discussed. A fast CELP algorithm for speech compression is studied as well.

These ideas, along with the others presented here, will give you the ammunition to push the boundaries of digital communication. Pushing boundaries is a ham tradition, one we need to nourish as much as possible. Don't simply read the works of these authors, go to your keyboard or workbench and starting *doing*!

As in previous conferences, all papers in these proceedings are unedited and solely the work of the authors.

David Sumner, K1ZZ
Executive Vice President

Newington, Connecticut
August 1994

Table of Contents

A Proposal for a Standard Digital Radio Interface

Jeffrey Austen, K9JA
2051 Clearview Drive
Cookeville TN 38501, USA
Internet: jra1854@tntech.edu
PacketBBS: k9ja@wa4uce.#midtn.tn.usa.noam

Introduction

Just about everyone who has ever used packet radio has had to deal with what should be a simple task: that of properly connecting radios and terminal node controllers (TNCs) together. Unfortunately, many people have learned that it is not very simple. Not only do the proper signal connections need to be determined between each radio and TNC but the correct audio levels must be set in order for the system to work well. This problem is compounded for persons with multiple TNCs or multiple radios. Every time a radio or TNC is changed, the system must be readjusted for proper receive and transmit audio levels, as well as proper delay times to accommodate the key-up time of the transmitter. These problems are exacerbated by the existence of differing connectors for different models of radios and TNCs. All of this can be attributed to the fact that the interface between the equipment uses analog signals despite the fact that packet radio is a. mode of digital communications. For operation at speeds greater than 1200 bits per second (b/s) most radios do not even provide a connector for the appropriate signals. Operators of digipeaters or remote sites are burdened with the task of hauling around extra test equipment and adjusting radios on-site instead of performing these adjustments in a convenient location such as a laboratory or home station. Emergency operation is difficult because it is almost impossible to properly connect various equipment quickly in the field unless the exact configuration is known beforehand.

In this article a proposal for a digital radio (DR) interface is developed. This interface is designed to support all current packet modulation methods and speeds and any which can be reasonably anticipated for future use. It provides “plug and play” operation between any digital radio and TNC (from here onward the term TNC refers to a TNC or any other device, such as computer or packet switch, which processes the data being communicated). It can be easily incorporated into most of the current equipment and it allows the use of a single radio in multiple packet modes without changing cables or making any adjustments.

Requirements

The requirements of the interface are as follows:

- connect any DR to any TNC;
- be transparent to the data stream;
- operate over a wide range of speeds;
- operate with both synchronous and asynchronous modulation modes;

- operate in both full- and half-duplex modes as well as in transmit-only and receive-only systems;
- provide good immunity to electromagnetic interference (EMI);
- be tolerant of variations in the equipment: not require any adjustments when equipment is changed;
- operate over cable distances from zero to at least 10 meters;
- be usable for all existing digital communications modes and for all anticipated modes in the future;
- operate at all existing speeds and at all reasonable future speeds, at least up to 2 Mb/s;
- have a single standardized connector so that connection is "plug and play;"
- sense when cable is disconnected or when the DR is powered down;
- make use of existing standards, where possible; and
- allow easy migration from the current system.

Development of the Interface

In a digital communications system the digital information is communicated by representing the information as an analog signal. For the interface the simplest representation should be used for the information being sent: this is a serial bit stream. To do this, it is necessary to move the "modem" out of the TNC and into the DR This change has a benefit of making the dual use (voice and data) of the radio easier to accomplish. A front panel switch could easily select between voice and one or more data modes; for example, a 2-m radio might be built to support voice, 1200 b/s packet and 9600 b/s packet.

Many standards have been developed for use in data communications. Some standards which are related to the needs of this interface are EIA/TIA-232, EIA/TIA-422, EIA/TIA-423, CCITT V. 10, V. 11, EIA/TIA-449. There appears to be no standard which provides the necessary functionality; however, some standards can be incorporated into the interface.

Examination of the information which must be communicated across the interface yields the required signals. The fundamental information which must be conveyed across this interface is receive data and transmit data. Auxiliary information is necessary to indicate where each data bit is, when the data is valid, and when the data can be sent and received. To accommodate both synchronous and asynchronous systems at varying speeds a synchronous interface is used. The receive and transmit clock signals originate at the digital radio and are independent of each other.

To send data from the DR to the TNC the following items are necessary:

- Receive Data: the data from the DR to the TNC.
- Receive Clock: a clocking signal for the receive data, originating at the DR.
- Receive Data Valid: a signal originating at the DR which indicates that the receive data signal is valid (similar to carrier detect).

To send data from the TNC to the DR the following items are necessary.

- Transmit Data: the data from the TNC to the DR.
- Transmit Clock: a clocking signal for the transmit data, originating at the DR.
- Request To Send: a signal from the TNC to the DR indicating that data transmission is requested.
- Clear To Send: a signal from the DR to the TNC indicating that data transmission may proceed.

One other signal is necessary to convey the DR status to the TNC.

- DR Ready: a signal from the DR to the TNC indicating that it is powered up and capable of reception and/or transmission of data.

The Interface Proposal

The signals listed above will be sent using a combination of EIA/TIA-422 (differential) and EIA/TIA-423 (single-ended) signal levels. The two data signals and two clock signals, because of the potentially high speed will use differential signaling, which provides for speeds of up to 10 Mb/s. These signals will use eight wires of the interface. The status signals will use single-ended signaling because high speed is not necessary. These signals will use four signal lines and two ground lines (one in each direction, per EIA/TIA-423 specifications). To insure proper operation under fault conditions (either unit is powered down or the cable is not connected) "fail safe" line receivers must be used for the four status signals (RDV, RTS, CTS, and DRR).

Much of the delay necessary at the beginning of the transmission are due to internal delays in the transmitter. This delay is made the responsibility of the DR rather than the TNC. When CTS becomes active, data can be sent immediately; after the last bit of data has been sent, RTS may become inactive. Additional delay may be added in the TNC (as is done currently).

The physical connector selected is a high--density 15-pin D-series connector. This connector is small enough to be used on mobile and portable equipment and yet it reasonably rugged, reliable and inexpensive. The male connector (plug) is used on the TNC and the female connector (socket) is used on the DR. Although the same type of connector is popular for computer displays., the opposite sex connector is used on the computer so that confusion should not occur. Cables will. act as "extension cords," that is they pass all pins straight through from the connector on one end to the other end. The shell of the connector must be used for the shield connection if a. shield exists on the cable; if no shield exists the shells must be connected by a wire in the cable, All DRs and TNCs must have metallic connector shells so that shielded cables can be used effectively.

Alternative Interconnections

Although the interface is specifically designed to connect a DR to a TNC, it can be used to connect two DRs or two TNCs together or it can be used in a transmit-only or receive-only system. To connect two DRs together there needs to be a adapter which contains a FIFO large enough to accommodate the largest packet at the maximum speed differential between the systems. To connect two TNCs together there needs to be an adapter which generates appropriate clock signals. The receive and transmit signals are independent of each other so they can be running at different speed

or be going to different DRs; use in a transmit- or receive-only system is also possible (a protocol other than AX.25 will be necessary in this case).

Incorporation Into New and Existing Equipment

This interface can be incorporated into new radio designs by including the "modem" with the radio and providing a method for switching modes (e.g., voice, 1200 b/s data, 9600 b/s data). Most TNC designs can be updated quite easily to incorporate the interface without eliminating any current features.

Most existing systems can be easily modified to use the new standard. It appears that the PackeTen, DataEngine, PI, PI2 and PackeTwin cards and Kantronics DataEngine modems require very little modification as the appropriate signals are available to easily add the interface; the only significant change is that with the new interface the modem is physically housed with the DR, not the TNC. In general, any modem which performs clock recovery can be easily modified for use with this interface. Any TNC which provides the "modem disconnect header" can have the interface added to it by using that connector. A smooth transition from the current system to using the new interface can be made by providing adapter kits for common modems and TNCs so that current equipment will not be obsoleted.

Summary

The interface proposal presented here will solve the problem of connecting digital radios and terminal node controller or computer equipment together. It provides a simple, inexpensive, versatile, and easy-to-use solution. It is applicable to all current packet radio systems, as well a other digital systems and it does not inhibit future improvements to packet radio systems, either in the modulation and coding techniques or in the protocols. While the exact specifications remain to be finished and tested through implementation, much existing technology is being used and no problems are anticipated. The author welcomes suggestions for the improvement of this interface and is interested in hearing from a few persons who are willing to design and test interfaces for various modems and TNCs.

Automatic Packet Reporting System (APRS)

Bob Bruninga, WB4APR

Introduction and editing by Gwyn Reedy, W1BEL

Introduction

The Automatic Packet Reporting System (APRS) is a multi-purpose program for the PC which makes use of data from the amateur packet network to provide a number of interesting and valuable functions.

APRS embodies WB4APR's experience over the last 13 years using packet radio for real-time communications in public service events. It also incorporates the capability for operating over non-local distances without use of the existing packet network.

APRS accomplishes the real-time display of operational traffic via packet broadcasts and map displays.

Historically, almost every aspect of HAM radio communications has as its root, the interest in the location of other stations. Look at DX maps, countries worked, counties worked, grid squares, mobile chatter; everyone is quite interested in where other stations are.

Secondly, APRS avoids the complexity and limitations of trying to maintain a connected network. It permits any number of stations to participate and exchanges data just like voice users would on a single voice net. Any station that has information to contribute simply transmits it, and all stations receive it and log it.

Packet radio has great communication potential but so far has been best used for passing large volumes of message traffic from point to point or into the national distribution system. It has been difficult to apply packet to real time events where information has a very short life time. Typically, several steps are involved in preparing and passing message traffic including decisions about routing and connectivity.

APRS recognizes that one of the greatest real-time needs at any special event or emergency is the tracking of key assets.

-Where is the Event Leader?
-Where are the emergency vehicles?
– Where's the fire?
– Whats the Weather at various points in the County?
-Where are the power lines down?
– Where is the flood?
-Where is the head of the parade?
-Where are the VIP's?
– Where is the mobile ATV camera?
-Where are the mobiles?
– Where is the hurricane'?

With the advent of affordable Global Positioning System (GPS) receivers, the powerful capabilities of APRS are greatly enhanced for the public service community (especially the Civil Air Patrol and disaster management", boaters, hikers, etc.

The latest versions also provide specialized support for DX cluster system users, and the direction finding capabilities are a boon to 'fox hunters.'

APRS should not be considered just a 'mapping program.' While it has a powerful capability to use vector maps (including USGS 15° grid and CAP 7.5°grid maps, and gridsquares) to display the position of objects, the program uniquely and powerfully handles databases such as country callsign prefixes, National Weather Service sites, airport locations, etc. NWS information may be automatically via a landline dialer function. Headings to any object may be calculated and displayed.

Display Screens

There are three major display subsystems and a number of other minor displays.

Latest Beacons

This display maintains a list of the latest UI frame received from each station. In effect, this is a multi-station one-line broadcast message system. Since the lines contain the LATEST time of receipt, this display shows if a station is still on line within the last few minutes.

Positions

This display maintains a separate list of the positions of each station. Each position report can also contain a brief comment. These lines show the latest time of receiving a given position report and give an indication of the latency in the network over unreliable paths such as HF.

They also contain Beam Heading for Direction Finding, and Weather conditions for weather reporting stations.

Maps

Maps to any scale from 0.125 miles up to 20,000 miles can be displayed. Stations are instantly displayed when they transmit a properly formatted position beacon. Stations with a reported course and speed are automatically dead-reckoned to their present position. A complete database of all the National Weather Service stations is built in. You can

center the map anywhere in the world.

Traffic

In addition to the BEACON text which is used to broadcast information to all other stations on the net, there is an operator-to-operator message capability. Any station can send one line messages to any other station. On receipt, the messages are acknowledged and displayed on the bottom of the receiving stations screen until the operator hits the K key to kill them. These messages are ideal for station-to-station communication while remaining within the APRS environment. However, they are not as efficient as the connected protocol, and should not be used routinely for Rag-Chewing on a busy APRS net. To rapidly exchange text, got to Talk mode and connect to the guy.

Read Mail

This screen shows the last 23 lines of messages exchanged by any stations on the net. Is useful for "READING THE MAIL".

All Traffic Log

This display is a time sequenced log of every new beacon or one line message sent. Beacons are logged the first time they are received. This is in contrast to the LATEST display which shows the most recent time of receipt of a beacon text.

Heard Log

This display maintains a count of the total number of transmissions from each station per hour. These statistics are ideal for displaying the connectivity of the network over varying paths, such as HF, or to see when stations enter and leave the net.

Digipeater List

This Display shows the full raw packet header so that APRS users can see what digipeater paths are being used by other stations.

The proper use of digipeaters is important in an APRS network.

Station Tracking

Although APRS automatically tracks mobile packet stations interfaced to GPS or LORAN navigation, the graphic capability of the maps works perfectly well with manual tracking or with gridsquares. Any station on HF or VHF that includes his gridsquarein brackets as the first text in his beacon text will be plotted by APRS. Additionally, any station can place an object on his map including himself and within seconds that object appears on all other station displays. In the example of a parade, as each checkpoint with packet comes on line, its position is instantly displayed to all in the net.

Whenever a station moves, he just updates his position on his map and that movement is transmitted to all other stations.

To track other event assets, only one packet operator needs to monitor voice traffic to hear where things are. As he maintains the positions and movements of all assets on his screen, all other displays running APRS software display the same displays. The Tracking command on the P display will cause APRS to keep the map display always centered on a selected object.

Grid Squares

APRS now also plots stations by gridsquares. Since four-digit grid squares only locate a station to the nearest 60 miles or so, and six-digit gridsquares only specify stations to the nearest 3 miles or so, APRS will not display stations reported via gridsquares on map ranges less than 128 and 8 miles respectively. Stations reported by grid squares will each be assigned an exact LAT/LON which is offset from the center of the grid according to an algorithm based on the letters of their callsigns. This prevents all stations in the same grid square from all being displayed on one spot in the center and spreads them out in the grid The resulting POSIT in the POSITION list is annotated to indicate that the position is approximate.

Another advantage of GridSquare reporting in APRS is that it allows cautious people to participate in APRS without revealing their exact location. It is also very brief. Six characters vice seventeen. This is an advantage when reporting via MIR or SAREX.

Space Applications

APRS could be a solution to the effective use of orbiting terrestrial style packet radio digipeaters in space such as on the Shuttle, MIR, AO-21 and ARSENE.

The problem with space digipeaters is the saturation on the uplink channel which makes the use of a normal CONNECTED protocol impractical. For a CONNECTED contact, a total of five successive and successful packet transmissions are required

Not only does APRS reduce this to one packet, but it also capitalizes on the most fascinating aspect of the amateur radio hobby, and that is the display on a map of the location of those stations.

If all stations were encouraged to simply insert their LAT/LONG or Grid Square as the first characters of their beacon text, or even better, a compressed form of their location in the 'TO' field of the Unproto command, everyone within the satellite footprint would see the location of every successful uplink.

Since the shuttle is a rapidly moving object, the locations of successful uplink stations will move progressively along the ground track.

All it would take to implement this capability is a single AMSAT news bulletin to ask all stations to insert their POSITS in their beacon text or

Unproto string. No changes onboard the shuttle or MIR would be required.

(Ed comment: One additional change is perhaps some attitude adjustment on the part of the user community to be more accepting of digipeated UI frames. This too is a valid form of communication deserving of access to the spaceborne systems.)

Fox Hunting or Direction Finding

APRS is an excellent tool for plotting the location of a hidden transmitter, balloon, or interfering signal. APRS will display the intersection of bearing lines from a number of reporting stations. To use APRS in this manner, each station having a bearing report on the direction of the target simply enters that bearing using the OPS-BeamHeading command His station will then not only report his location, but also a line of bearing. All stations running APRS can simply hit the X key to display the intersection of these bearing lines. Further, if a DF vehicle has a GPS or LORAN device on board, he can be tracked and directed right to the location of the target. There is an optional Doppler DF registration for direct connection of a Roanoke or Doppler Systems DF unit for automatically plotting andtransmitting instantaneous DF bearings. Please note that APRS uses 360 degrees for North and 000 to indicate that no direction information is available.

Weather Station Reporting

APRS position reports can also include the wind speed and direction, as well as other important weather conditions. APRS supports a serial interface option to the ULTIMETER-II home weather station. With this interface, your station includes WX conditions in your position report for display at all other stations in the network All weather stations show up as a bright blue circle, with a line indicating wind speed and direction. Remember that APRS uses 360 degrees for North and uses 000 to indicate that no wind direction is available. Each of these stations can be highlighted in turn with a single key stroke, so that all WX reports across the state can be had at a glance.

APRS also has a database of the locations of all the NWS sites in the USA for instant display. APRS can also crunch a file of NWS hourly WX conditions and update all NWS stations on the map.

Using Dumb Terminals In An APRS Network

The simplicity and usefulness of this geographic capability cannot be over stressed. Stations running APRS simply move the cursor to where they think they are on the screen and their LAT/LONG coordinates are automatically transmitted to all other stations.

Even the simplest of portable packet stations with dumb terminals can report their positions if a pre-printed map is made available to all net participants which has a LAT/LONG grid reference. The portable station just looks at the map and enters his LAT/LONG into his beacon text. Using the same map, he can plot with pins the location of all other stations as he sees their position reports go by. APRS also plots station positions based on Grid Squares. Eventually, we hope that all stations, no matter how they are using their TNC, will include their LAT/LONG or Grid Square in their Beacon Text so that their location is immediately available.

DX Cluster System Monitoring

APRS will grab all DX SPOTS and put them on the ALL list and maintain a list of all DX cluster users on the local node in the LATEST list. And finally, APRS will plot the DX spot by callsign prefix or Gridsquare if given as the first four letters of the comment field!

Chessboard

To demonstrate the flexibility of APRS in reporting the movement of objects on screens in a net, I have drawn a chessboard map in the center of the Gulf of Mexico. Any two stations can play chess easily using APRS by placing pieces on the map using the INPUT-ADD command and updating their positions using the cursor and INSert keys!

Monitoring stations that have also zoomed into the chessboard will see the game progress too! You should consider going to an unused frequency so as not to clutter an active APRS net.

Protocol Techniques

Since the objective of APRS is the rapid dissemination of real-time information using packet UI frames, a fundamental precept is that old information is less important than new information. All beacons, position reports, messages and display graphics are redundantly transmitted but at a longer and longer repetition rate. Each new beacon is transmitted immediately, then 20 seconds later. After every transmission, the period is doubled. After ten minutes only six packets have been transmitted. After an hour this results in only 3 more beacons; and only 3 more for the rest of the day!

APRS version 5.06 implements a decay mechanism which adapts to channel usage.

APRS Digipeaters

To satisfy the objective of instantaneous response, APRS stations are designed to begin operating without any prior knowledge of the network. For this reason, all APRS stations are initialized with the alias of RELAY and to send all UI frames via the path of RELAY. With this form of generic alias callsign (RELAY) and wildcard digipeating (RELAY), a mobile, or new station

on the air does not have to know anything about the network in advance, but to simply turn on his computer to be seen by adjacent nodes.

Although digipeaters work poorly for AX.25 level 2 connections, they are ideal for APRS operation using UI frames only.

The minimizing of wildcard addressing and multiple repeats when not needed is the key to an efficient APRS network.

In the Washington DC area and Chesapeake Bay area, we are establishing a network of WIDE area digipeaters on the simplex packet frequency of 145.79. This frequency is for Keyboard QSO's and all UI frame applications. Even leaving personal mail boxes on the frequency is welcome, since mail is posted at keyboard rates and is read off-the-air by the mailbox owner without QRM. The normal CONNECTED operation of BBS's, mail forwarding, file transfers, TCP-IP and DX clusters is discouraged!

Wildcard Digipeating

To make these WIDE area digipeaters respond to mobiles and new stations, all wide area digipeaters have the same alias of WIDE in addition to their normal FCC callsign. This second generic alias of WIDE adds tremendous flexibility to APRS networks by significantly extending the ranges for wildcard digipeating using well situated permanent digipeaters.

These wide area digipeaters are spaced several tens of miles apart so that they are not too close, but that they can hit their adjacent other WIDE digipeaters.

Assuming WIDE area digipeaters are about 30 to 50 miles apart it is very easy to select an UNPROTO path prior to a road trip which will assure that your location packets will always get back to your home area. The following example shows a string of digipeaters along the east coast. The HAM calls of SOUTH and NORTH are used for clarity.

CALL: NORTH-3 NORTH-2 NORTH-l HOME-O SOUTH-l SOUTH-2 SOUTH-3

ALIAS: WIDE WIDE WIDE RELAY WIDE WIDE WIDE

If the mobile is going south for the day, and will be operating in the vicinity of SOUTH-3 digipeater, the operator can preset his UNPROTO path to be via WIDE,SOUTH-2,WIDE.

Notice that not only will his packets make it back to home from the area of SOUTH-3, but also from the area of SOUTH-l since SOUTH-l will also respond to the first WIDE in the list. Similarly, stations in the vicinity of SOUTH-3 are alerted to his movements as he leaves home, since the WIDE,SOUTH-2,WIDE specification is symmetrical. If he set the UNPROTO path to SOUTH-3, SOUTH-2, SOUTH-l in the usual manner, he would not be tracked at his homeuntil he actually arrived at his destination.

As you can see, having the flexibility to alternate the generic aliases of RELAY or WIDE with other known sites gives a good degree of flexibility without having to change the UNPROTO path while on the road. Using the three digipeater string, he can wander up to 150 miles in his planneddirectionandstillbetracked by the XYL. If he has no idea where he is going, he can always use the path of WIDE, WIDE or even WIDE, WIDE, WIDE and go anywhere, but with greater QRM on the channel. Yes there are multiple collisions, and repeats, but the packet does get out to the third tier!

Pre-emptive Digipeating

The ultimate APRS digipeater configuration is to have modified digipeater code so that any digipeater hearing a UI frame with its callsign anywhere in the UNPROTO path will pause for a reasonable time andthen digipeat the packet as long as it was not previously digipeated by any stations earlier in the list,

This way, to always report your movements back home, you always place digipeaters in your UNPROTO command in the reverse order of your travels. Your packets will be digipeated back to your home area as you enter each new digipeater in your direction of travel. For example, if you live in the vicinity of DIGI-1 below and routinely travel in the direction out to and including DIGI-3.

DIGI-1 DIGI-2 DIGI-3 etc.

The mobile could specify the UNPROTO path of VIA DIGI-3, DIGI-2, DIGI-1 in order to be tracked anywhere all the way out to the area of DIGI-3.

If only DIGI-1 hears the packet, it will pre-emptively digipeat the packet and set its digipeat flag.

If DIGI-2 also hears the original packet, DIGI-2 will pause for P seconds to see if DIGI-1 repeats it. If so, it does nothing, since DIGI-1 follows it in the list. If not, after P seconds, it digipeats the packet for DIGI- 1 to subsequently further digipeat in the normal manner.

Similarly, DIGI-3 pauses for 2*P seconds to see if DIGI-2 digipeated the UI frame. If so, it does nothing. If not, after the 2*P seconds, it digipeats the packet.

Even if the packet pauses and comparisons are not performed, (to simplify the code) the worst case is that N duplicates will arrive at the destination for all N digipeaters that simultaneously heard the original UI frame. Since these are UI frames, any pauses in the network for the comparisons suggested are not sig-

nificant. The extra code to do the pauses and comparisons only protects against duplicates when two digipeaters hear the same original packet.

This algorithm works perfectly well in reverse. If a mobile desires to announce his progress forward in the direction of his travel he can specify the digipeaters in the forward direction. Then using this algorithm, all of his packets will be repeated in the forward direction, no matter where he is along his route, but not in the backward direction.

Until we get new UI forwarding algorithms, the general aliases of WIDE and RELAY will do nicely. If fixed, known digipeaters are available, even with the generic alias of WIDE, it is best for fixed APRS stations to use the digipeaters unique callsign instead of alias to avoid any ambiguity. Also avoiding the wildcard addresses except when necessary, significantly reduces QRM on the channel.

APRS now has a special command that sets ones own station to the ALIAS of WIDE vice RELAY. This is so that an APRS station that is well situated, can serve as a WIDE digipeater. This command should be used with caution and with the understanding of all stations on the net. Too many WIDE's and too close together causes too much QRM.

PacComm also added a new UI frame in their 3.2 ROM so that the POSITION information would be independent from the BText. This LText is just like the Beacon Text, except it is a separate entity with its own timing. This keeps the BText free for other applications. (particularly, for announcing WHAT your mobile is doing, and what symbol to use, etc....) This maintains the same distinction between BTEXT and POSITS that APRS already handles easily.

Similarly, the LText command allows you to manually enter your LAT/LONG or grid square in your TNC, even without a GPS, so that TNC's in networks will send their locations periodically. The LText permits a free text format so that it is compatible with any future specific formats (currently APRS parses GGA, RMC, VTG, APRS L/L, PACCOMM and grid squares and a future 8 character compressed L/L format) and there will probably be others too.

Network Considerations

Since NODES are so much smarter than digipeating, the ultimate solution is to have the NODES do all UI frame routing. The APRS station simply sends his UI frame TO APRS VIA HOME; Any NODE hearing that transmission that has knowledge of the route to HOME, will send the single packet via the NODE network (internode, level 4) to the HOME node! When it arrives at the HOME node, it is transmitted once as a UI frame. With this arrangement, a mobile only has to specify his one intended destination, no matter where he travels!

DIGI/NODE COMPATIBILITY: Since the user should not have to change his digipeater path as he drives from one area to another, he should be able to specify a path that is compatible with both nodes and digipeaters. This is easily accomplished by assuming that the LAST field in an UNPROTO digipeater list is the HOME NODE and should be the ultimate destination for the UI frame through any level 4 network. Any and all preceding fields are assumed to be digipeaters only.

With this arrangement, the user could use an UNPROTO path of APRS VIA WIDE, HOME so that any generic WIDE digipeaters would repeat his position to their local area as would any WIDE NODES in the usual digipeater fashion. Only the node that hears the direct packet would also forward it through the network at level 4 to the HOME NODE. If only one field is included in the digipeater string, it would be interpreted as both a digipeater and a HOME destination without any difficulty. Digipeaters and NODEs would digipeat it, and nodes (hearing it direct) would forward it at level-4. It is important that NODES hearing digipeated UI frames from other digipeaters do NOT enter the packet into the network, to eliminate duplication. Only the ones hearing the direct signal should be responsible for doing the level 4 routing.

EXAMPLE: A typical mobile just wanting to keep his spouse informed of his whereabouts might want to just use the UNPROTO path of APRS VIA HOME. Then his UI frames will be digipeated by the local HOME node or digipeater and will also be routed back to HOME by all NET- NODES along his travels. If he also wants to be seen by most HAMS in the areas of his travels, then he sets his path to APRS VIA WIDE,HOME. If he travels through a region that has both digipeaters and NODES, he might choose APRS VIA WIDE,WIDE,HOME. This way any areas with digipeaters would digipeat via. WIDE,WIDE and if he gets to an area with nodes which are aware of the path to HOME, then they will forward his packet there.

Finally, since I hope to build a regional area Tracking network, the node should also permit the SYSOP to turn off other level 4 routing if he wants to make a dedicated network of APRS nodes just for tracking. Such a network would be swamped if all of the: BBS and other CONNECTED protocol users began to use it, and the original purpose of the network would be defeated.

Still, most of these APRS support ideas could be included in all NODES so that a minimum of APRS

tracking could be supported by ALL networks on all frequencies, especially where there is not yet a dedicated APRS TRACKING NETWORK I think there are other undeveloped applications for shipping UI frames through ALL networks which have not yet been explored. The capability should be there, in any case, so that experimentation can proceed.

Using APRS for Space Communications

The Automatic Packet Reporting System could be a solution to the effective use of orbiting terrestrial style packet radio digipeaters in the amateur satellite program. To date there have been three AX.25 1200 baud FM transponders flown in space. The first was on the Space Shuttle STS-35, the second was on the space station MIR, and the third has been via the FM transponder mode of AO-21.

The problem with a space based digipeater is the total saturation on the uplink channel which makes the use of a normal CONNECTED protocol impractical. For the SAREX robot QSO mode, a total of five successive and successful packet transmissions were required to constitute a successful contact. Of an estimated thousands of uplink stations, only about 250 were successful.

Recognizing the stringent requirements for success using the CONNECTED protocol, provision was also made to recognize those stations which were successful in getting only one packet heard onboard the shuttle. Over 700 stations successfully completed single uplink packets.

APRS takes advantage of this unconnected, one packet mode to demonstrate successful uplinks to the shuttle. In addition, however, it capitalizes on the most fascinating aspect of the amateur radio hobby, and that is the display on a map of the location of those stations.

RV And Mobile HF Net

We have begun a nation wide Boat, RV and APRS position reporting net on HF using 7.085 and 10.1515 MHz LSB. (Yes this is in the band! It is the same as saying 10.147.1 USB, but the convention on HF is to specify packet frequencies using the LSB convention.)

All boaters and Recreational Vehicles are welcome! To see the locations of all stations on the net, tune your TNC to the exact frequency and monitor for at least 15 minutes. When you first activate APRS, it will send out a query to all stations on the network for their positions.

Packet Positions On All Frequencies

Encourage all BBS's, NODES, Servers, and stations in your area, to start placing their LAT/LONG in their beacon text using the format: BT!DDMM.xxN/DDDMM.xxW/... .comments. (In order to accept this data in TheNet ID beacons, APRS will accept this position format anywhere in the ID text. APRS will also plot the positions of stations reporting by Grid Square surrounded by brackets [FM19xy]. If all packet stations get in that habit, then APRS will automatically plot a map of packet activity on any frequency!

Objects

As noted previously, anyone may place an object on the map and all other stations will see it. In their systems, on their P-list, the object's position report will be marked with thelastthreelettersofthestationthat is currently uplinking that position to the net.

A neat feature of APRS is that any station that has more current information on the location of that object can update its position by hooking, moving the cursor, and then hitting the insert key. Now this new station begins uplinking the new posit, and all stations, will update their P-list entry for that object INCLUDING THE ORIGINAL UPLINK STATION! The new position overwrites the old one so that the original station will now no longer uplink it.

This cameinhandy during hurricane tracking. Who ever had information on the latest NWS EMILY position, uplinked it and everyone then always saw the latest storm track without anyone in the net being dependent on any one station for updates!

Once objects are transmitted on to other station map screens, they will remain there until that operator deletes them. Even if the original station stops sending the object position, it will remain there forever. Once the object or station has not been heard from for 2 hours, it will fade to gray so that you know it is an old contact. In version 4.0 1 a feature was added so that you can suppress the callsigns of old contacts. Just press the J command, and select LATEST instead of selecting any specific object type.

The result will be to redraw the map showing ALL symbols, but only the calls of the recent ACTIVE stations less than 2 hours old. Another feature added recently is the KILL function. This permits the uplinker of an OBJECT to KILL it from all displays on the net. His station will continue to uplink the object, but tagged with a special KILL flag to suppress its display on all screens. It remains in everyone's P-lists, though, so they can refer back to it if needed. They must still manually DELete it from their P-list as needed.

Load Sharing

Since any station can take over reporting of any objects, one approach is to let only one station hook every symbol that comes in and then he becomes the reporting responsibil-

ity. The original station that uplinked the report in the first place will fall silent when it sees the report coming from the designated Net Control station. This way all positions are reported by only one station on frequency, although all other stations can still update the positions as needed. Remember that the last station to report the position of an object will be the one that continues to report it!

Propagation Statistics

A secondary benefit of the redundant beacons is that it operates like a poor-man's chirp-sounder. Since APRS keeps statistics on the number of packets heard from each station over the last 24 hours, this display can be used to verify HF connectivity between stations throughout the day. It's like a free-bee radio check every 15 minutes everywhere! After watching APRS statistics for just a day, or so, the daily variations in propagation conditions to all stations is visible at a glance. Further improvements in connectivity is possible by changing frequency bands during the solar day. By saving statistics on each band in a different file, the APRS user can use this data to optimize his connectivity at any time of day or location.

Weather Reporting

All stations on the net can be apprised of unusual weather conditions by any station placing a weather symbol on the map. Just like stations, weather symbols will be dead reckoned between reports. In this way APRS is ideal for reporting the movements of hurricanes and tropical storms. There are over a dozen different weather symbols for this type of weather reporting. Secondly, APRS has an optional interface routine for automatically reporting the wind speed, direction, temperature and rainfall from the ALTIMETER-II home weather station. All stations with this interface show up on the maps as a large blue DOT with a line indicating the wind direction and speed. Their position report also includes the temperature and the rainfall. Similarly, any station can select to use the Weather station symbol for his station, and can manually enter his wind speed and direction for display on the net.

Waterway Net Operations

It is recommended that all Waterway Net participants that are HF packet capable begin reporting their positions on the HF APRS nets. No changes to the existing voice net on 7068 are required! Since APRS will be operating continuously, 24 hours a day, it will provide a reliable and continuous background reporting of most stations. This will free up the voice net for passing of more voice traffic, and for position reports from non packet stations. One APRS station should volunteer daily to uplink the voice position reports into the network from his display by placing them on his screen as OBJECTS. Once these reports are being uplinked into the APRS net, any other APRS station can assume reporting responsibility for that OBJECT (station) simply by uplinking a later report. If the original station uplinking an OBJECT hears a later report, it will update its screen with the new report and will no longer report on that OBJECT since another station has taken reporting responsibility for that OBJECT. This enables stations to pass off APRS reporting responsibilities and keeps the network from being dependent on specific full time stations.

Differential Correction

Tom Clark (W3IWI) has installed a Differential GPS transmitter in the Washington DC area transmitting 30 second DGPS data on the APRS frequency. APRS GPS mobiles can now obtain accuracies to 5 meters or so. We are pleased to report that the RTCM- 104 format works perfectly well with APRS and with TNC's.

Although this is an excellent demonstration. and there are surely HAM applications that can take advantage of the DGPS accuracy, APRS is probably not one of them. First, APRS is not concerned with NAVIGATION accuracy, because a) no maps are that accurate (with DGPS you can make 'em so!), and b) the purpose of APRS is to inform others of mobile locations over a wide VHF area, NOT to the nearest 15 feet. (APRS formats do maintain positions to 60 foot precision) Secondly, A mature APRS net involved in a special event. or activity, can probably NOT handle the QRM from 30 second RTCM transmissions.

In the long term, the DGPS data should probably be transmitted MORE OFTEN and on another frequency, OR be remotely controlled such that it can be requested by a mobile user on demand, but silenced most of the time. Transmitting less often is meaningless due to latency of the data The only application of DGPS that I can think of is to keep track of golfcarts at a hamfest, and be able to see who's booth they are at. I will probably incorporate a ?RTCM? format in APRS to permit stations to request DGPS data.

Broadcast, UI and un-connected protocols - the future of Amateur Packet Radio?

Paul Evans W4/G4BKI, Tarpon Springs, FL.

Abstract

Anoverview of the user applications of packet radio is presented and the relevance of connected protocols in each scenario is examined. Suggested formats for unconnected systems are made and conclusions drawn.

Introduction

The increasing explosion of levels of data in the amateur network sometimes defies belief. Luckily, the amateur network (within obvious financial constraints) adapts at the local level to match each "crisis" in data throughput as it is reached. However, how long can this continue when spectrum is limited by bandplans and the necessity of keeping modes apart? Is the increasing use of ideologically "unsound" commercial wormholes for amateur use because of another reason? A deeper look at the efficiencies of spectrum use has to be taken and the inescapable fact is that protocols are the over-riding factor in channel efficiency, with modem design and TX/RX switching times probably being the next most important in turn.

Uses of packet radio

The vast majority of amateur packet usage can be sub-divided into the following applications:

1) Networking - end-to-end linking using ROSE, TheNet, TexNet, etc.

2) BBS - providing store and forward message service

3) PacketCluster - semi-real-time DX information and round-tabling

4) TCP/IP - robust connectless technology

5) APRS - real-time position and information system

6) End-user - keyboard to keyboard chatting

The order in which these applications are presented is a best estimate as to the amount of on-air time they occupy. Naturally, there is cross-fertilization between some of these areas - an end-user puts a message on a BBS and it forwards it to @WWW. The end-user did very little, the BBS "layer" did some more, but the network put it across the whole world - creating thousands or millions of packets in the process.

By way of further analysis, let's go through each one of these major applications to see where improvements could / should be made. The comments here are intended to be constructive and not cynical or demeaning of any of these applications which, after all, have appeared and adapted to suit each of their niches in turn. That is not to say that they have been implemented in the most efficient way, however.

1) Networking

The arena of networking is polarized into two schools of thought - ROSE and (the predominant) TheNet in it's various guises. As we know, the two work in distinctly different ways in routing data, with TheNet using the NET/ROM protocol of nodes broadcast in order to selectively adapt to changing RF routes. There is much to be done here in improving the network layer - such as using signal strength as part of the selection process in establishing path qualities. It is the author's opinion that ultimately, for better or worse, the nodes broadcast with embedded additional information in some form will win out in this area. Adaptive networks are the answer in RF data linking. ROSE does adaptation around "broken" paths, but not on the fly, it does it by selection of pre-programmed alternative routes.

Furthermore, it is increasingly common to see TCP/IP (sic. IP) data traveling on the ROSE and TheNet networks. Why? See TCP/IP below.

2) BBS

BBSes were the first real application of packet radio, after simple end-to-end keyboarding. Why did it start? You have to guess that it was because the typical amateur didn't want to burn telephone time talking to landline BBSes. However, much of the information by way of messages, files, etc. is repeatedly put over the air time after time in connected fashion to end-users. Monitor the traffic on the output of a BBS for long and you'll see this immensely inefficient use of spectrum. Similarly, the inter-BBS traffic has been pretty inefficiently handled. In the early days a more-or-less random forwarding system was used, at least nowadays hierarchical forwarding (in which the author was an instigator with G4MTP in the UK back in 1988) and data compression techniques are in place during forwarding.

3) PacketCluster

Several years after BBSes came PacketCluster, just when amateur BBS technology was maturing and obtaining a high degree of stability. Those of us who have a long term interest in HF saw this as the first "REAL" application of packet radio. It has to be said that the effects of PacketCluster have been widespread. Mainly because the data has to be timely, PacketCluster requires a fast network, with good reliable

connectivity. Luckily, the owners of many of these systems have a lot of time and MONEY to invest with the final goal being only one thing - timely delivery of spots. Unfortunately, much of this investment is wasted because of the connected protocol used throughout the system. Just total up all the packets sent in the European or US-wide PacketCluster when one spot is entered. It is thousands. Multiply that by the number of spots entering the system during a contest weekend (probably one every 10 seconds) and it's no wonder that spots appear 10 minutes late - the whole network is creaking under the load.

4) TCP/IP

TCP/IP has enjoyed it's advocates (and critics alike) ever since it appeared on the amateur scene. The critics have mistakenly thought that it was a "channel hogger", stealing time from AX.25 This is not the case. It is simply a reflection upon the more efficient channel usage algorithm encoded in IP than in AX.25 Unfortunately, because TCP/IP is executed on an external PC and requires a higher degree of user input, there has been a mental picture that this must be for packet "gurus", with, let it be said, some resentment at times. When the protocol becomes closer to the average ham in the form of elements of the code in a TNC, then maybe this unfortunate distinction will no longer appear to exist.

The chief advantages of TCP/IP are:

a) it is commercially compatible. No other protocol in the amateur scene can so readily "hook" to DOS and UNIX applications

b) it uses connectless technology with (theoretically) nothing to time out and with on-the-fly re-routing. It IS efficient, because only frames get repeated over long distances if they are genuinely missing from a sequence

c) the protocol is very robust - it almost has to be by definition - it is not necessarily timely.

One of the difficulties of early usage of TCP/IP was building an entirely separate inter-connected TCP/IP network of stations. Now that ROSE and TheNet transport IP frames, the requirement for a different network has vanished. What is the most efficient long distance means of tranporting data now ? IP over network code.

5) APRS

Most amateurs in the USA will by now have heard of Automatic Packet Reporting System (APRS) software, written by Bob Bruninga, WA4APR. This consists of a full GPS position reporting network based entirely around UI broadcast frames. It's simplicity is the heart of it's functionality and possibly it's explosive growth in the past few months. Reporting stations give position reports or messages as UI frames (the messages have their own end-to-end acknowledgment protocol within the frame but are sent as UIs), while APRS stations periodically broadcast known targets in a similar way to TheNet. Provided there is a continuous end-to-end string of APRS stations, objects local to an APRS station in, say, Tampa can be seen hundreds of miles away in, say, Tallahassee. Every APRS station shares the data available.

The bottom line for APRS is that it is now a MASSIVE network nationally, and yet it still uses only one 2m frequency. Now that is spectrum efficiency!

6) End-user

Keyboard-to-keyboard chatting has virtually died it's death, maybe because hams are fascinated by talking to automatic gizmos. In fact, in some instances keyboarding which has been seen going on by a node Sysop has (mis-guidedly) resulted in those end-users being given limited access to the node! There are attempts in the South-East USA to produce an end-user network, but whether that will fully function with any great efficiency has yet to be seen. Keeping other traffic such as BBS forwarding and PacketCluster off clean, "pristine" bits o:f network is extremelv difficult to do (conversely, I know just how much money and effort went into keeping the UK PacketCluster backbone free of "unwanted" traffic). Unfortunately, with high loading connected technology, there will always be some element of antagonism between different user groups. At the end of the day, end-users are dependent upon some or all of the above systems to get their data through, unless of course it is simply one-on-one across town (in which case their packets could be lost on virtually any packet channel without distress or complaint by anyone).

Discussion

Possibly UI frames got a bad reputation because they were seen in the early days of packet as simply a means to provide Beacon frames. When users were desperate to see anything that was going on in t-heir area, beacons and use of the Mheard list were the only answer in AX.25. Very soon, however, the word was out that beacons were to be avoided at all costs other than at "key" network sites for identification. Their use fell dramatically at that point. Without question, the only use of unconnected streams from that point was TCP/IP.

The fly in the ointment that now raises great questions is, undoubtedly, the emergence of APRS. It has turned amateur connected technology thinking completely on it's head. Even with existing simplistic UI framing, APRS has proven that over a very doubtful HE path the positions of sailing yachts reported over an entire summer had a success rate of over 20% (and by extrapolation around 80% on VHF). The weakest area of APRS is having to

work around existing non- optimized AX.25 UI protocol and that network code hasn't been able to transport end-user UI frames in connected format end-to-end. This is something that is sure to be resolved shortly.

It's clear that somewhere this channel efficiency was noticed by somebody. At the time APRS started it's growth spurt, around the InterNet went messages bemoaning the in-efficiency of PacketCluster. Why all those frames carrying the same message over and over to different addresses and often on the same frequency that the inter- node data came in on (horror of horrors, a flat data network!)?

Solutions

Clearly, we can see by now that there is a (very) common thread running through ALL of the above and ALL applications in the amateur packet environment. What's needed to fix the problems and how much will it help?

1) Use connect-less robust technology wherever possible.

2) At all possible network nodes, use the instantaneous RF route quality to reflect the true quality of the path(s). Evolve the most efficient means of dissemination of this routing information to neighboring nodes/switches.

3) Either change all inter-BBS traffic to broadcast addressed TCP/IP (255.255) or provide an alternative connectless piecing together of data frames on a shared basis (if I'm a BBS, I'll assemble as much of a message as it takes before I either accept or reject the message as something I will need to have or not).

4) All network code to have the ability to forward connect-less traffic transparently.

5) Modified UI frame to allow "ping" function. One frame is put out by a new (assumed to be highly mobile) user. Others answer in turn so that a complete local overview is built of channel activity.

6) Modified UI frame which allows information to either be broadcast in the "normal" fashion or addressed to a number of (listed) stations. Each station responds with an ACK. Retries are made by the "base" until the "delivery list" is clear or until time-out occurs.

7) Provide improved local network control using intelligent software with the ability to use either mobile GPS derived position information or previously broadcast fixed position information, in addition to RF routing quality information.

8) FEC type frames to provide more robust broadcast technology.

9) The ideal network is one which manages itself and adapts to suit the changing circumstances in which it operates. One reason why commercial and amateur networks fell apart after the 1994 Southern California earthquake was that either complete re-building of the network was required (skillful) or the latency of existing paths was too high. In times of system "panic" a heavy load "flood" algorithm should be available to re-establish routing as required.

Note: In future it looks like the position information of a station will become paramount in network control. There is an explosion in the use of Maidenhead locators being used by stations in UI frames (of the format [AAnnBB] with the square brackets). PLEASE, if you do send beacons for E's or tropo work, use this format.

These new "features" could have a very significant effect upon the requirement for data bandwidth. Let's look at a typical example:

As it is:

Take the situation where PacketCluster Node W6GO has 20 end-users and is linked on HF to VE3DXC with 15 end-users and the East Coast Cluster with, say, 20 cluster nodes and a total of 200 users (a not too uncommon situation).

User W6DU puts a single spot into W6GO over a two hop path. This gives 6 frames. W6GO then "creates" a minimum of 40 frames of activity with local users. Two frames (minimum) occur across the HF link. VE3DXC now creates another 30 frames locally and two more to the East Coast (in a DIFFERENT frame format too!). The East Coast cluster bursts into action and creates a total (minimum) of around 440 frames. In the end, 520 frames have been generated to support 235 users, NOT counting the numerous re-tries that there WILL be over all that distributed network OR taking into account any additional networking between nodes or from nodes to end-users.

As it "should" be:

User W6DU puts in the same spot. Users local to him use that information immediately. This one FEC frame (or 3 times repeated UI frame) tells W6GO there is a new spot. W6GO simultaneously sends one FEC frame which is heard by ALL users (either true end-users or the pseudo-user VE3DXC) IN THE SAME FORMAT. VE3DXC immediately does the same single frame FEC to his end users, etc. The number of frames is therefore reduced to a minimum of 25.

The result: a 20 fold decrease in spectrum occupancy of all types. Just stop and think. Why did you just spend all that money going to 9600 Baud dedicated PacketCluster links when 1200 Baud will do it easily? Because the protocol used forced you to. Just think what 9600 Baud will REALLY do for you.

Adding this new data efficiency in the real world.

This could be done two ways. Firstly, build a new special version of code to implement just the new protocol (potentially useful for dedicated, high speed sites). Secondly, build expanded code which would fit in with the existing network, providing a new "super-set" of functions. To do that it must first be built around what's out there in use - the TNC-2 clone. One problem which has dogged the code writers at all the TNC-2 manufacturers has been one of CODE space. For some time now, modem TNC-2 code really has been a quart being squeezed into a pint pot. This is an area where no single TNC manufacturer has been prepared to boldly break the mold of TNC-2 compatibility on the Z-80 platform, and for some very good reasons. Fortunately, the network code authors (notably Dave Roberts, G8KBB, author of TheNet X1.52) have provided a standardization route here. The most frequently used alternative code in a TNC-2 is network code and so it is logical that anything that works there is workable for all. The development of the bankswitch (using a firmware driven SIO line to switch between the lower and upper page of a double size EPROM) has suddenly extended the life of the Z80 TNC-2 and all of it's compatibility way into the future. This now makes available all that code space which can be used to provide new code power as well as backwards compatibility with "pure" AX.25. The way forward appears to an ad-mix of some elements of network code, IP and end-user code residing within the end-user TNC. Digipeating was always in the end-user code, why not move more level 3/4 "smarts" into the end-user TNC and build routing tables and connectless protocols on the expanded platform?

Faster processors in the TNC-2 platform (such as the new PacComm 20MHz processor Backbone version of the SPRINT-2 TNC) will further allow more exotic code to be run before the TNC-2 design runs out of life.

Future platforms will certainly have to be built around next generation micro-processors which are ideally suited to real-time data I/O tasks, either the 28018 1/2 or 68000 series. Much of the existing Z80 code could be used with the 280182, whereas 68000 code would need to be built from scratch (although the Gracilis route from TCP/IP and AX.25 compatibility in NOS is certainly attractive here).

Despite these dreams, the TNC-2 compatible, it can be assumed, will be with us for a long while yet and that it where the code will probably be targeted in the first instance.

Has this realization process been started?

The answer, to limited extent, is YES! The process was started in a discussion between the author and WA4APR in which the amount of data emanating from PacketCluster was mentioned and so were some ways in which more generalized improvements in data transport might be realized. It has to admitted too that many commercial applications would benefit from such improvements. Very soon Bob had incorporated code into APRS which would monitor PacketCluster traffic, throw away unwanted information, monitor inter-node traffic, etc. Basically give you information the moment it becomes available (which, you'll recall, is what it's all about,). Expanding upon this, stations including Maidenhead locators where instantly placed on a correctly scaled map. Finally, the DX spots themselves are placed on the map as they come in.

Future improvements may be to create multi-user "switches" using connectless technology and to use position information for network building and control . The applications are general, but PacketCluster is certainly going to be one of the chief beneficiaries of this approach.

Furthermore, the TNC code side is currently under development to provide enhanced end-user connectless technology. Once these areas of supervisory (IBM-PC) code and TNC code are more fully developed, one can expect it to start displacing or molding the design of existing systems around it.

The bottom line

Existing data dissemination systems deliberately steer away from shared data. Why is APRS so efficient? It works around a COMPLETELY flat, data SHARING system using ONE channel. It really does make you think "where did we go wrong?".

Conclusion

Without some major re-thinking of protocols, the amateur network will forever fight un-necessary data bottlenecks. For the foreseeable future, data rates will be limited by the availability of suitable radios. Until the day that high speed radios are readily available, the main improvement to the network will be made in the area of protocol. More specifically, protocol which has been adapted to the needs of the applications most favored by the end-users. As practically all the systems provide one-to-many services in one form or another, this is the area where work would be best focused. There is still a place for connected technology, but more and more it looks like unconnected technology is the future of amateur packet.

Packet, GPS, APRS and the Future

Paul Evans, W4/G4BKI, Tarpon Springs, FL.

Abstract

The Global Positioning System produces wider ranging applications every day. The relatively easy hook-up to amateur radio devices makes it the ideal experimenter's "toy" and tool. The benefits it can provide the world of amateur radio are numerous, not least of which is the way in which it brings that element of experimentation back into the hobby AND impresses the "authorities" whenever it is demonstrated or used in disaster situations.

Introduction

GPS and packet radio - experimenter's paradise!

The advent of inexpensive Global Positioning Satellite receivers means that many applications either not thought of or out of reach previously are now becoming possible. The focus of activity has shifted from simple stand-alone GPS receiver units available at first, to the use of remote data gathering in conjunction with radio links. The introduction of the differential GPS service (applying local correction data over a low frequency radio link) to improve accuracy seems to have been the trigger which is causing experimentation with GPS and data links. One thing is very clear - like video recorders and CD players before them, GPS receivers will soon be with us ALL in one form or another. Even the industry itself probably doesn't realize the impact it will have.

Getting it together

Amateurs were perhaps a little slow to exploit the scope of GPS applications, but this is excusablein view of the initial cost of GPS receivers. Now that several GPS engine manufacturers have produced OEM (board level) receivers, cost is reduced and experimentation is enhanced (the previously hidden features of the cards can now be accessed). In addition, very aggressive marketing by one particular GPS manufacturer in the "consumer" nautical market has resulted in obsolete (but none-the-less usable) units being available for several months at "rock bottom" prices. The knock-on effect is that the next generation of hand-held, plotting, differential ready, miniature receivers have hit the market at even lower prices!

One of the more frustrating things about modem systems is that protocols are hard coded into hardware/firmware i.e. on the serial links to many units, very particular command structures have to be used and a specific flow of data and acknowledgment is needed. This makes hooking units together very difficult for the kitchen table experimenter. Luckily, a substantial number of amateurs have the hardware and software available to them to be able to write software which can provide this "connectivity". More importantly, because their programming time is substantially "free" many hours can go into code writing without it becoming an impossibly expensive task.

Now that amateurs have their hands on GPS receivers in increasing numbers they are moving quickly in this area and most significantly they are working on disaster awareness programmes with high levels of mobility/portability. A major spur to this is that it is exceptionally useful during a disaster preparedness exercise AND the real thing, to know where your roving helpers are without having to ask them all the time. The typical ham HT can be used for both the voice communications and the GPS data gathering.

The major amateur experiment is being spearheaded by Bob Bruninga WA4APR and his Automatic Position Reporting System (APRS) software. In this, the original communication with the GPS receiver was using one of the PC serial ports and communicating position data out via packet radio by using a packet TNC attached to a second serial port. The main reason for doing it this way being that, as we said above, interfacing the data directly between the output of the GPS and the input of the TNC was tricky.

Some important information and the "down" side of GPS data.....

There are numerous GPS Receiver output formats, the simplest to handle being pure ASCII strings and the NMEA-0183 format. Additional complication occurs because the serial leveling may be either TTL or RS-232 whichcancause some interfacing problems. Also, the baud rates can be different, mostly 4800 Baud is used right now, but 9600 Baud is also common too. When purchasing a GPS receiver the buyer must be very careful to make sure that all these factors are taken into account. Unfortunately, the receivers are marketed as black boxes and the sales people at your local store are extremely unlikely to know what you're talking about, or worse still, guess with plenty of "yes" answers in order to make a sale! Very few manufacturers put data of this detail on their product sheets. When they do it is not to be trusted. A very famous receiver product brochure picked up recently actually listed the output strings it would produce - the only problem was that in practice it was wrong!

There are numerous output formats, rates and physical types. They are

broken down (mostly) into the following:

Serial: TTL (OEM units generally) or RS-232 (most consumer units).

Baud rate: 4800 (generally) or 9600 (sometimes).

Format:NMEA-018 1, NMEA-0182, NMEA-0183, binary, TSIP, etc.

$GPGGA - Gives HMS, Lat, Long, #satellites, Altitude.

$GPRMC - Gives HMS, Lat, Long, Bearing, Speed, Date.

$GPLLA - Gives Lat, Long.

$GPVTG - Gives Bearing, Speed.

There are some overall guidelines for the sentence structure: $GP is for GPS, $LC for LORAN and proprietary strings from manufacturers, normally starting with a short form of the manufacturers name ($MG is for Magellan, etc.) are used.

The industry seems to have gone out of it's way to spread a web of confusion over every possible aspect of output data. It rivals the RS-232 "standard" Video2000/BetaMax/VHS and UNIX fiascoes in it's complexity. One can't help feeling that in several years there will be a process of "re-standardization" just to make things work! Fortunately, there is enough commonalty left in the system to provide workable solutions for the amateur experimenter and hopefully it will stay that way.

One threat is that NMEA propose to adopt a licensing/royalties system for use of it's format - this is bound to push the industry into completely proprietary (and arbitrary) formats.

Unless the industry giants form a close knit circle and co-operate there exists a real danger that life will not be so easy in the future in matters GPS.

The "up" side of GPS data.....

Fortunately, TNCs are now available which eliminate the need for a PC in the system. In their latest guise (firmware version 3.2), all PacComm TNCs contain a GPSTExt command into which can be placed any string that a parser should look for. That string may be $GPGGA, $GPRMC, or indeed anything (allowing forward compatibililty and also the ability to work with LORAN strings too). When the presence of that text is seen, the parser loads the whole string into the LTEXT, CTEXT and STEXT with periodicity of the Location beacon set in the same way as normal beacon text is handled The BTEXT is left available to beacon out any other useful beacon text required. Furthermore, the LPATH command sets the destination for the LTEXT UI frame (the default is GPS, but may be changed to APRS, etc.) including digipeat path(s).

Certainly, for now, the method of using a parser in the TNC code is the simplest route to reducing the amount of air time required for each position report. It is used to throw away about 95% of the typical output from the GPS receiver. It also makes sure that it doesn't matter what the periodicity of the output data from the GPS receiver is (if it can be set), a frequent problem in early experiments which limited receiver choice or increased channel clutter.

Differential GPS over the air

Taking the technology to the next step naturally entails the provision of a more accurate system using differential GPS. Differential base stations are expensive, but there are several appearing on 145.79MHz. The good news is that the serial output from the differential base station is simply "piped" into a TNC and allowed to broadcast. The differential equipped GPS receiver at the other end does all the work, throwing away the packet header and swallowing the data wholesale. What looked originally like a tricky problem actually works like a dream!

One very important factor for the amateur world to consider is that the provision of the US Coast Guard MF DGPS service is being looked upon increasingly as an expensive (to the user) and potentially unreliable service. Likewise, other commercially available systems using encryption and sub-carriers on FM broadcast stations look like an expensive subscription "cable TV" type option. That commercial users are buying 2m receivers and TNCs and are eavesdropping on the APRS channel is a certainty. Suddenly, that puts amateur radio in a different light for a lot of people as a real service provider in. both emergency and normal times.

Conclusion

There :is no doubt that GPS and APRS have caused a great stir in a very short time. The commercial world has it's eyes on this technology, and well it might because it is so powerful. Many TNCs are out there simply listening to what's going on on 145.79MHz, many in the hands of non-amateurs who cannot believe what they see! It gets rarer these days to see-amateur radio really at: the forefront of technology, but this is certainly attracting attention!

Computer Networks in Africa:
From Utopian Discourse to Working Reality

by Iain Cook

April 8, 1994

for Lorna Roth

International Communications

Media Studies

Concordia University

"We are moving toward the 21st century with the very great goal of building a Computopia on earth, the historical monument of which will be only several chips one inch square in a small box. But that box will store many historical **records, including; the record of how four billion world citizens** overcame the energy crisis and the population explosion; achieved the abolition of nuclear weapons **and complete disarmament; conquered illiteracy; and** created a rich symbiosis of god and man without the compulsion of power or law, but by the voluntary cooperation of citizens to put into practice their common **global aims."**

Yoneji Masuda
The Information Society

"I don't think the battle lines have **been** drawn, but it's a real mistake to link ail **the approaches and attitudes to this new technology into one. If you listen to** Tim Leary, he talks as though this will bridge cultural gaps. That's the most unmitigated bullshit This stuff has no relevance to any issues in the Third World. It's completely arrogant and self-indulgent and has nothing to do with science."

– Bill Buxton
Computer science professor
University of Toronto

The positions outlined on the previous page are diametrically opposed. On one hand, a hopelessly utopian vision is painted of the computer as saviour of all humankind. On the other, a very sceptical outlook is enunciated, of the potential for change enabled by computer networks in the developing world. I believe that there is room for a middle ground to be claimed between these two opposing positions. This terrain would acknowledge both the benefits and limitations of computer technology in the developing world. However, in order to venture into this space, it will be necessary to look at what other kinds of discourses are being enunciated in relation to computer technologies.

The task of this paper is to explore the less-utopian discourses surrounding computer networks in the developing world, and contrast these discourses with the everyday practice of computer networks. In particular, this project will look at how one computer network, RINAF, is set up in Africa. The information for this investigation has been culled from various sources across the Internet. The bulk of the information has been provided by men and women working in the field.

* * *

In 1992, the United Nations Conference on the Environment and Development (aka, The Earth Summit) was held in Rio de Janeiro. Out of the conference came the Rio Declaration on Environment and Development: Agenda 21. Agenda 21 is the "non-legally binding authoritative statement of principles for a global consensus on the management, conservation and sustainable development" of the earth's resources. (The Rio Declaration on Environment and Development, 1992)

In relation to computer networks, Chapter 40 of Agenda 21 urges governments, the UN system, and NGOs to:

> exploit various initiatives for electronic links to support information sharing, to provide access to databases; and other information sources, to facilitate communication for meeting broader objectives, such as the implementation of Agenda 21, to facilitate intergovernmental negotiations, to monitor conventions and efforts for sustainable development to transmit environmental alerts, and to transfer technical data.
>
> [RDED, 1992]

The use of computer networks is seen as a vital tool in the strengthening of the capacity for 'traditional information'. This term is not defined in the Agenda 21 document, though the document (chapter 40.11) does exhort UN member states to,

> with the cooperation of international organizations,. . . establish supporting mechanisms to provide local communities and resource users with the information and know-how they need to manage their environment and resources sustainably, applying traditional and indigenous knowledge and approaches when appropriate. This is particularly relevant for rural and urban populations and indigenous, women's and youth groups.
>
> [RDED, 1992]

The picture drawn is a compelling one. Computer networks are to be set up to enhance grassroots participation in a global communications network. It is convincing rhetoric such as this which seems to oblige thinkers in the development field to write:

> The large proportion of the analysis and recommendations devoted to electronic networking clearly demonstrates an unprecedented understanding by the Earth Summit, that we are not dealing here with just another technology, but with a substantial reformulation in the way people deal with each other on a global scale.
>
> (Bissio, 28)'

This 'substantial reformulation' seems to involve a revolution in the way power relations are to be considered in the coming years: "If information is indeed power in the present-day world, decentralization and networking are the new synonyms for the old utopia we call democracy." (Ibid, 30)

The same type of utopian rhetoric can be seen in the writing of Hamid Mowlana. Mowlana (1993) identifies four broad areas of socio-cultural change that occur as a result of computer networks and networking in the area of transborder flow: decentralization of decision-making and control; technological and cultural heterogeneity, decrease in hegemony, and weakening of the nation state and the public sector.

In these four categories, Mowlana sees the dynamic of computer networks' benefits as not being completely unmediated. For example, he sees a tendency towards centralization and consolidation of the policy planning and organization of organizations and institutions. However, he also argues that the formation of international networks "can allow for a greater participation of separate entities in the decisions and direction of the

---

* Though in my exposé of the utopian rhetoric surrounding computer networks in the developing world, I am to quote only 2 writers, Roberto Bissio and Hamid Mowlana, their writings are representative of the idealistic discourse surrounding computer networking and development. Both writers are renowned in the field of communications and development. Bissio is Executive Director, Instituto del Tercer Mundo (Third World Institute), Montevideo, Uruguay; Mowlana is currently Professor of Communications and Director of the International Communication Programme, American University, School of International Service, Washington, D.C.

organization and work in the direction of decentralization." (Mowlana, 24)

As well, Mowlana recognizes that the ability to connect a large number of organizations around a single purpose can result in a certain level of homogenization, both at the technological and cultural levels. This trend towards homogenization is countered, he argues, by the diversification of networks by the inclusion of voices from different contexts and cultures, which leads to an overall level of heterogeneity of communications flow.

As for the question of hegemony, Mowlana sees hegemonic interests being threatened by computer networks whose objectives include empowerment and democratization. Mowlana argues that as the number of informed users increases and as these users become part of the decision-making process, the result is a democratization of the power structures.

The fourth dimension of change engendered by computer networks, the weakening of the nation state, results from the ability of computer networks to work outside traditional lines of state power and administration. These networks are often created parallel to the state's existing communication infrastructure, and are of a non-governmental nature. Therefore, Mowlana argues, there is a potential weakening of the nation state and of the public sector.

At this point, what I would like to do is examine Mowlana's map of the four areas of socio-cultural change, in relation to a real-life computer network project, the RINAF project in Africa. I have chosen RINAF (the Regional INformatics Network for AFrica) as my site of study as it is a UNESCO-sponsored project (funded by the Italian government) and as

such, is the kind of computer network envisaged by Agenda 21. * The aim of the RINAF project is to:

- use new information and telecommunications technologies to favour exchanges between African countries;

- remedy the isolation of development and research institutions in African countries and facilitate dialogue between researchers, academics and industrialists;

- develop an operative process for the coordination, integration and upgrading of African networks, as well as exchange with other international networks.

(L. Abba, S. Giordano, S. Trumpy, 1394)

Though RINAF has specific aims which differentiates itself from other development projects in Africa, the way in which RINAF operates, on a technological level, is similar to other projects.

* There are other computer networks operating on the African continent by a broad range of different organisations: quasi-state bodies like the Centre for Scientific and Industrial Research in Accra, Ghana; international non-governmental bodies such as the UN Economic Commission for Africa in Addis Ababa, Ethiopia; single large International NGOs like Environmental Liaison Centre International (ELCI) in Nairobi, Kenya; coalitions of NGOs as in MANGO in Harare, Zimbabwe; and autonomous NGO service organisations such as WorkNet in Johannesburg, South Africa. [Mikelsons, 199 2]

The technology used by RINAF is that of FidoNet. ** Because of the history of the development of FidoNet***, FidoNet technology has been touted as the determining factor in making computer networks decentralized and democratic. The following excerpt from the American Association for the Advancement of Science/African Academy of Sciences Workshop on Science and Technology Communication Networks in Africa, held in Nairobi in August 1992, is indicative of the technologically determinist rhetoric surrounding FidoNet technology:

> Yet electronic networking, especially the decentralized Fidonet network, which is predominant in Africa, has spread in an essentially grassroots, i.e., non-hierarchical, manner, from user to user. The -bottom up- approach has been central to the character of Fidonet, and periodic attempts to impose a -top down+ organizational structure upon it have met with successful resistance.
>
> (Schoneboom, Gimbel, and Levey, 1992)

** The FIDO network (called FIDONET) is an extremely wide spread network based on store and forward capabilities for the transfer of files, texts and mail. Each node has a modem which lets users send or receive mail by logging in, without any charge. Each user can also send or receive files from the node. To give to every user the opportunity to use the network the FIDONET node assigns each user a maximum time limit, after which the user is cut off by the system. Access to the system is in this way similar to that offered by BBS (Bulletin Boards Systems). FIDONET is organized in several regions corresponding to continents. In each region there is a node with the task of storing and forwarding the traffic produced by the entire continent to the other continents. The transmission of mail and files can usually be achieved by the multiple storing and forwarding of files from a system to the nearest one reducing the cost of the calls performed by each system.

***FidoNet is known as the "people's network. In 1983, Tom Jennings, a computer programmer, began working-on-bulletin board software that would provide a link between the east and west coasts of the United States using homegrown bulletin boards. His scheme was loosely patterned after the amateur ham radio operators' network. A feature of the Fido software is that individual bulletin board operators can agree to a regular automated exchange of messages between their systems. This results in a web of linked Fido bulletin boards spanning countries and continents. This is collectively known as FidoNet.

This passage begs the questions: exactly how is the FidoNet technology implemented in Africa? Are there local conditions in Africa which might change the way in which FidoNet technology is used?

The limitations of computer networking in Africa may be considered as being characterized by two factors: technological and bureaucratic. Arni Mikelsons' (1992) recognition of these technological drawbacks is grounded in the understanding that Africa operates with a poor telecommunications infrastructure which is not conducive to the development of data communications. Telephone densities are as low as 0.1 percent of people. In the vast rural areas, where 80 percent of the population lives, there is often no access to basic telecommunications facilities.

Specific barriers include:

- poor switching equipment which results in incompatible signaling systems between countries;

- unreliable primary power sources which cause numerous telephone traffic disruptions;

- lack of trained staff to maintain the networks and keep abreast of newly emerging technologies;

- inappropriate management information systems for monitoring operational status of the networks;

- insufficient funds for system management and maintenance;

- continuous deterioration of lines due to climatic conditions for which the equipment was not calibrated by the manufacturer.

In light of these technological limitations, Mikelsons identifies four main reasons for choosing Fido in Africa:

- the technology optimizes the use of few, low quality phone lines;

- FIDO software is relatively inexpensive and often free for non-commercial use, running on all low cost computers frorn floppy disk PC to Macintosh Classic;

- it is becoming very easy to use as programmers continue to refine it;

- its message format is such that information can move between FIDO and multi-line NGO, academic and commercial networks which dominate northern networks.

To a great extent, technological constraints have been minimized with the implementation of FidoNet technology. With error-correcting modems, messages and files are routinely sent successfully even over very noisy public phone lines, although transmission time may be slowed considerably, raising costs. (Schoneboom,Gimbel, and Levey, 1992)

And, where phone lines do not exist, radio technology can be used as information carriers. RINAF uses packet radio technology where phone carriers are unavailable. Packet Radio links form a network developed by volunteers from the world of CB and Radio-amateurs which modulate on radio carriers. (Abba, Giordano, Trump, 1994) In other words, instead of using phone lines as carriers for computer information, radio waves are used to transmit the same information from one station to another.**** Thus, it may be argued that technology can overcome the problerns of setting up

**** Packet radio systems, in both terrestrial and space environments, have the potential to provide the "missing link" of reliable and inexpensive communications from isolated regions. Integration with low-cost landline (telephone)- based systems could dramatically increase connectivity without significant increases in cost. (Garriot, 199 1)

computer networks in Africa. However, there are bureaucratic obstacles which stand in the way of successful implementation of such systems.

The development of computer networks is hampered by nation-states' telecommunications policies. They include policies regulating:

- the import and use of telecommunications equipment;
- tariff structures;
- network traffic and structure;
- the quality of service and management of telecommunications networks;
- regional and international cooperation of telecommunications development.

(Mikelsons, 1992)

The state regulatory bodies in Africa are the national PTT (Public Telephone and Telegraph) utilities. As Garry Garriot (199 1) reports, these state-run utilities must be considered when planning trans-national communications systems, especially those utilizing radio technology. In 19 86 it took the Relief and Rehabilitation Commission of Ethiopia, a government agency, more than a year to acquire temporary authorization for a three-week demonstration of packet radio in a CARE food program. After its success (the first of its kind ever), the government quashed all further experimentation and a permanent network was never implemented. Similar, though less dramatic, experiences have been logged in other African countries.

One international agency, while implementing a packet network in an African country, decided not to request authorization for its proposed packet radio system, but rather strategized a "fait accompli" situation under the guise of an existing voice radio license. When the system was ready for

inauguration, all the proper individuals were invited to a generous reception and any objection quietly buried. Time and time again, the difficulties and delays in licensing or obtaining temporary authorizations stymie packet radio projects in terrestrial and satellite applications alike. Every country and situation is different.

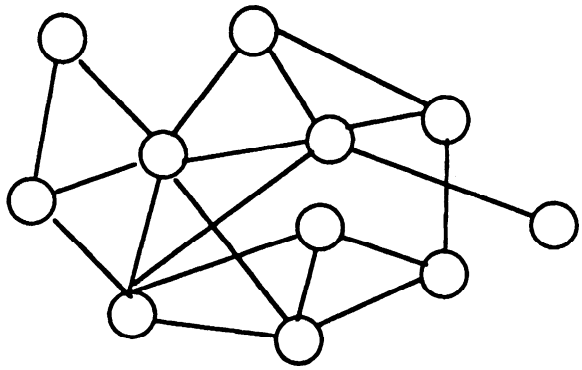
PTTs must also be considered at the local, more mundane level. Many African countries are now installing data line services, also called IPSS (International Packet Switched Services) which use an internationally standardized protocol for data transfer. The PTT – national post office or telephone company is almost always the operator of such a service and usually installs connection points to IPSS in the major cities. This service allows modem users in these cities to make a local phone call, and get on-line to any country with an electronic mail or database service.

To access such a service, the user orders a NUI (Network User ID) from the local PTT. A registration fee, a monthly or quarterly rental, and usage charges to connect to the remote host comprise the costs incurred for this service.

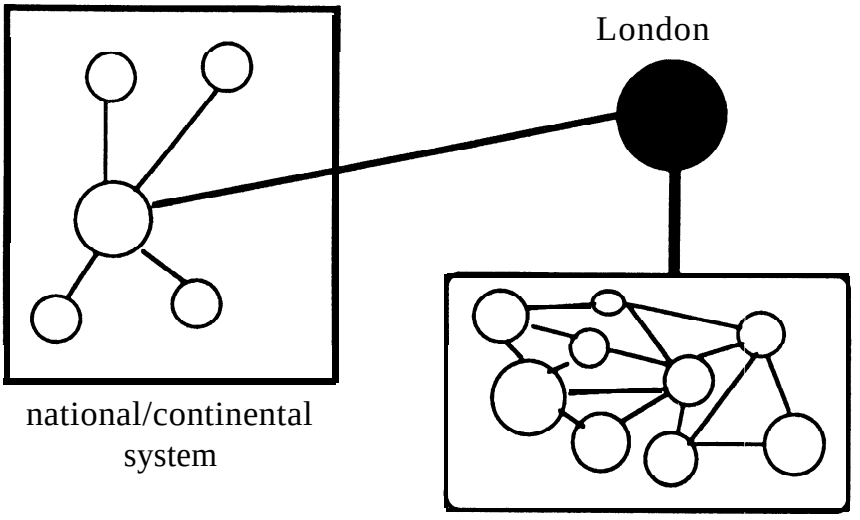
For regular computer network users, this NUI rental usually provides a significantly cheaper option than making a direct dial international phone call to the electronic host. If the host is accessed infrequently, then the cost of an NUI may not be justified. As with a normal telephone call, there is usually a substantially higher usage charge for connecting to a host outside the country than with a host computer inside the country However, since there are still very few mailbox host computers connected to an IPSS anywhere in Africa, there is really no option but to connect outside the country for mailbox service and pay the high rates, until one of the developing systems becomes connected to packet services .

Rate structures for IPSS are complex and vary enormously from one to country to another. Rental charges for a NUI can vary from $80 to $800 a quarter. Some PTT's require the user to rent PTT-owned modems at inflated rates. Even usage charges (which are based on time spent on-line and the volume of date passed down the network) can vary by a factor of two between different PTTs. Typically, the most significant portion for the charge is for the amount of data transferred. Users are charged both to send and to receive data, and this is frequently what makes the service prohibitively expensive. (Jenson and Sears, 199 1)

This intervention on the part of local PTTs is an important one, in regards to the organizational structure of FidoNet in Africa. Here's how a grass-roots FidoNetwork à la Jennings might be conceptualized:



However, with the intervention of the local PTTs, the dynamic of the structure changes, and begins to become less decentralized:



Because it is much less expensive to call into Africa rather than call out, phone calls originate from London.*** This characteristic of trans-national phone communications serves to change the dynamic of this form of communication. Information may flow in both directions, yet the decision of whether/when to initiate this data flow lies in the hands of the operator in London.

This is not quite the model envisaged by Mowlana. His promise of computer networks working to decentralize decision-making and control is undermined by the nature of this hierarchy. As well, there are cultural limitations which constrain the number of informed participants in these networks, and which weaken the threat to hegemony promised by Mowlana.

*** London provides a link to a global network for receiving or sending private messages and public bulletins. It is the site of a gateway operating at the Association for Progressive Communication's (APC) London host - GreenNet. Through this system users in Africa can gain access to the community of 10,000 NGOs and individuals working in peace, social development and environmental issues who use the APC network.

Garriot (1991) identifies a problem in the way in which networking technologies and information get disseminated in Africa:

> Affluent, liberal values emphasize information-sharing, while sharing is anathema in many traditional societies where information is power and the first to get and act on it becomes the most powerful. Information in this context is to be carefully guarded, because sharing it will only give someone else an advantage. To be sure, sharing does occur in such societies, but generally only within tightly-bound sub-cultural or familial groups and friendships. Thus, in an African context, it might be more natural to see fewer bulletin boards and more peer-to-peer links. There is, of course, a universal camaraderie among scientists and scholars that can mitigate somewhat the tendency to hoard information when the "pieces of the pie" – whether economic, academic or whatever – are perceived to be diminishing rather than increasing.

If Garriot is to be believed, then this question poses a serious threat to the promise of computer networks serving as agents of democratization at the grassroots level. If information is to be shared solely by academics and scholars, a less-heterogeneous, more-hierarchical model must be considered when mapping out the operations of computer networks in Africa.

In terms of Mowlana's fourth dimension of change – the operation of computer networks outside traditional lines of state power and administrations – we have seen how crucial the intervention of state-run PTTs are to the shape and dynamics of information flow in Africa.

Thus, I think that after a more-careful consideration of the actual operating practices of computer networks in Africa, it can be concluded that Mowlana's description of the four areas of socio-cultural change promised by computer networks is not quite precise enough. This imprecision can be attributed to a tendency to over-valorize the impact of computer technologies. However, this is not to say that Mowlana's analysis is not of any

value. I believe he has identified four key areas where traditional power structures can be challenged by the development of computer networks, especially where (relatively) low-technology applications, such as FidoNet, can be utilized.

Bibliography

Abba, L., S. Giordano, and S. Trumpy. "RINAF: A network interconnection project of academic and research institutions in Africa." Email sent by Wendy White, March 2, 1994.

Bissio, Roberto. "Integrated Information and Development Communication Networks." Development Vol. 3, 1993: 27-30.

Garriott, Gary L. "Packet Radio in Earth and Space Environments for Relief and Development." Paper presented at the 34th Annual Meeting of the African Studies Association, St. Louis, Missouri, November 23-26, 1991.

Jensen, Mike and Geoff Sears. "Low cost global electronic communications networks for Africa." Prepared for Panel on *Electronic Bulletin Boards and Computer Networks: Africa and African Studies in the Information Age*. 34th Annual Meeting of the African Studies Association, St. Louis, Missouri. November 23-26, 1991.

Masuda, Yoneji. "The Information Society." (Bethesda, MD: The World Future Society, 1981), p. 156.

Mikelsons, Arni. "Technical Report of the Global Networking Workshop." Toronto: Nirv Centre, April 1992.

Mowlana, Hamid. "Information Hunger and Knowledge Affluence: How to Bridge the Gap?" Development. Vol. 3, 1993: 23-26.

Ruth, Dr. Stephen R. and R. R. Ronkin. "Aiming for the Elusive Payoff of User Networks: An NGO Perspective." Paper presented at the annual meeting of the International Society for the Systems Sciences, Denver, Colorado, July, 1992.

Schoneboom, John. "Electronic Networking in Africa: Advancing science and technology for development." Paper prepared for the American Association for the Advancement of Science/African Academy of Sciences Workshop on Science and Technology: Communication Networks in Africa. Nairobi, August, 1992.

White, Wendy D. and Bob Barad. "FidoNet Technology Applications for Scientific Communication and Networking." Email sent by Wendy White, March 2, 1994.

A Low Cost DSP Modem for HF Digital Experimentation[1]

Johan Forrer, KC7WW

INTRODUCTION

This article describes an HF modulator-demodulator (modem) that is based on Digital Signal Processing (DSP) principles. A practical approach is shown, rather than the usual terse mathematics that usually accompanies this kind of discussion. This article is intended for those interested in experimenting with HF digital communications using DSP software. A low cost DSP platform is also described for implementing some of the ideas presented in this article including complete source code for a high performance HF digital modem.

What is DSP? To some, this means the manipulation of digital data to extract something meaningful. To the communications engineer, it actually means quite a bit more. Consider the following analogy: As experimenters, many are familiar with analog circuits that uses various interconnected components, such as resistors, capacitors, and operational amplifiers. The constructor uses some schematic or rather, an electrical behavioral model as a reference. Similarly, DSP in the most general sense, is the modellina of such systems in an all--digital domain, This involves sampling of real time signals where its accuracy, resolution, sample rate, as well as a multitude of algorithms plays an important role.

DEMODULATOR DESIGN

With that brief introduction, a little digression is necessary on the background of demodulation, in particular FSK (frequency shift keying) as used on the HF bands. It will become evident later, that this overview is appropriate for both analog and DSP demodulators.

After some experimentation with different types of demodulators, an experimenter soon realizes that the type of demodulator intended for use on HF is generally different than that used on telephone circuits, or that used on VHF. The main reason lies in the nature of the HF propagation. Not only has an HF demodulator have to deal with QRM, QRN, but also fading (QSB), multipath propagation, as well as with a very congested part of the RF

[1] For publication in Sept/Oct RTTY Digital Journal.

spectrum. To design a well-engineered HF demodulator, one must pay special attention to several key factors such as dynamic range, i.e. the ability to work with both very weak and/or very strong signals, Superior selectivity is also required to deal with adverse signal to noise (S/N) conditions.

The area of HF demodulator design has evolved over the years to an almost, universal arrangement. This becomes evident when analyzing the modems of current TNC's (terminal node controllers). This typical arrangement, or architecture, is shown in Figure 1.

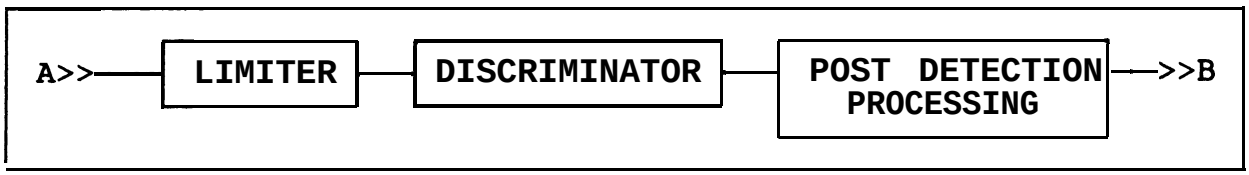


Figure 1. Conventional modem architecture. Audio input is at (A) and digital signal output is at (B).

The discriminator in these conventional designs, typically consists of a pair of filters, one each tuned to the mark and space tones respectively. The envelope of the outputs of the filters are extracted and combined as shown in Figure 2.

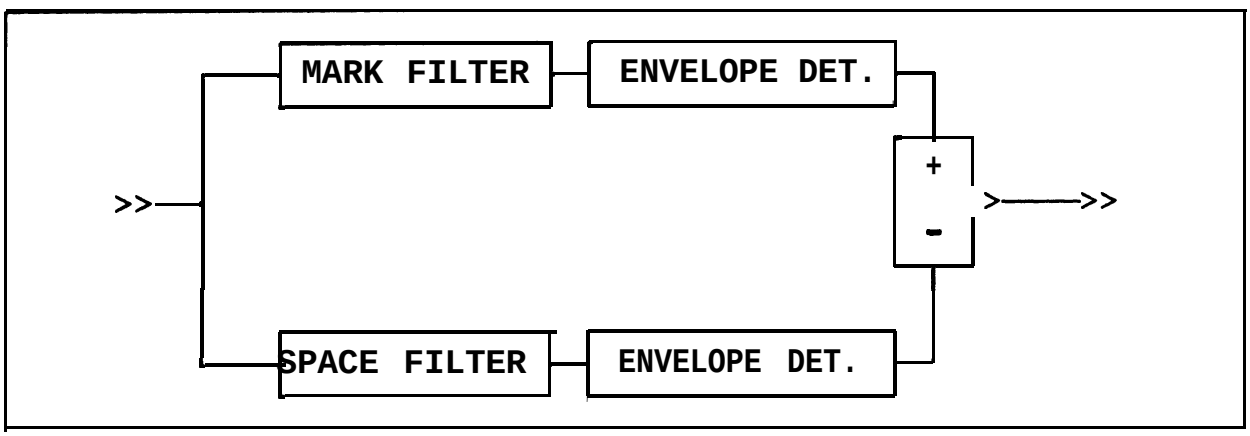


Figure 2. A conventional modem discriminator. Each filter is a bandpass filter centered at the appropriate tone frequency. The filter design loosely fits the specifications corresponding to a matched filter.

The operation of the demodulator is quite simple. The audio signal containing FSK tones is applied to the limiter at point (A) (please see Figure 1). The limiter produces a constant

amplitude, square-wave version of the input signal. One reason for this step is to remove amplitude variations on the FSK signal prior to detection. The function of the discriminator is to convert the input frequency to an analogous DC signal. It is obvious that, when, say a mark tone is present at the input, the mark discriminator filter will leave the mark signal unattenuated, but the space discriminator filter will attenuate the signal severely. The mark filter envelope detector will thus produce the equivalent of a positive envelope (a steady positive DC) and the space envelope detector will produce a near zero level, The combined output thus will be a positive DC level. Likewise if only a space tone is present, a negative DC level will be produced at the output of the discriminator. Such a discriminator will produce a classical **"S"** response when the frequency is swept between the mark and space tones where the upper part of the **"S"** corresponds to the positive part of the signal, i.e. the signal passing through the mark filter, while the lower part of the **"S"** corresponds to the signal passing through the space filter.

The remainder of the demodulator is concerned mainly with **post-** detection signal conditioning. I should be noted that, besides a DC level shift, the mark and space tone components, i.e. higher frequency components, are also present in the output of the discriminator. These tones are removed by a low **pass** filter. The remaining task of the demodulator is to threshold and convert the filtered DC levels to appropriate standard signal levels such as RS232 or TTL.

There are of course numerous variations on this basic theme, such as dealing with the efficiency of the various filters, balancing the outputs from the discriminator to track a small amounts of drift, or tolerate a limited amount of off-frequency operation.

In this discussion so far, the operation of the demodulator is nearly intuitively simple, however, this type of approach is what is known as a matched filter design, i.e. the detection of the tones is by means of filters that loosely matches the characteristic of the modulated tones. The reason why some demodulators perform better than others, even when the same architecture is used, lays fundamentally at the engineering principles of these matched filters. The theory of matched filters, is beyond the scope of this **article,** The **reader is** referred to the bibliography for further information.

Before we conclude this overview of demodulator architecture, several general, however, important observations must made. The first concerns signal phase. Note that any information regarding phase relationships within and between tones are of little consequence and plays no part in the detection process. When this is the case, the detection method is also known as noncoherent detection. One consequence of noncoherent detection is that it

requires somewhat more spectral bandwidth and also requires at least twice the shift magnitude than coherent detection. Its advantage is that its simple and cheap to implement.

The second observation deals with the usage of the limiter stage. This has been a controversy in the past. It was stated earlier that the purpose of the limiter was to remove any AM from the signal prior to detection. Its inclusion also has another purpose that has to do with non-linear characteristics introduced by the limiter. When a non-linear transformation of the input signal occurs, such as in the case of the limiter, the spectral content of the input signal is also modified. Theoretical research from the 1950's and 60's as well some practical evidence have shown that such a process perhaps may have desirable side-effects in signal capturing capability in the presence of strong competing signals (please see the bibliography for further references).

The third and final observation, also in context of the limiter stage, is dynamic range. Ideally, a demodulator should be able to cope with wide variations in signal levels such as often is the case when QSB is present or when dealing with weak signals. At one instance the signal level may be extremely low, then the next instant it may become very strong. When a limiter is used, its stage gain should be sufficient to handle all but the weakest of signals. Alternatively, as is used in this particular DSP modem, is to design a linear system with sufficient dynamic range so both weak and strong signals can be demodulated on an equal basis.

THE IMPLEMENTATION OF THE DSP DEMODULATOR

The DSP modem design that follows, employs several of the key components previously discussed in Figure 1, i.e. the matched filter detector and post detection processing. No limiter is used, however, special provision is made to increase dynamic range and provide additional improvement to the S/N ratio through a process of oversampling and decimation.

Most DSP applications involve analog to digital conversion (A/D) of the input signal. The rate at which the sampling and A/D takes place must be chosen rather carefully. As a minimum requirement, the sampling rate must be at least greater than twice the highest frequency present in the audio input. This is to prevent a phenomena called aliasing. The higher the rate the better, however, it must be kept in mind that the DSP must be able to complete its computational tasks associated with each sample before the next new sample can be processed. With first generation DSP's, this typically amounted to approximately 400 to 600 instructions for audio frequencies. If it is found that there is plenty of processing time to spare, a higher sampling rate may

be accommodated. This heavy demand on the amount of processing between input samples is one reason why general-purpose processors like the Intel 386/486 are not suited for DSP applications. Second generation fixed point DSP processors, like the TI 320C26 used in this DSP modem implementation, can quite easily accommodate even higher sample rates.

A further consideration concerning the choice of sample rate involves filter order. This factor may influence demodulator performance and usefulness. Larger filter orders at low sample rate generally mean **that** the signal lingers longer in the DSP and may thus introduce undesirable delays for timing-critical applications such as those in ARQ protocols.

Figure 3 presents the various components as employed in this DSP modem.

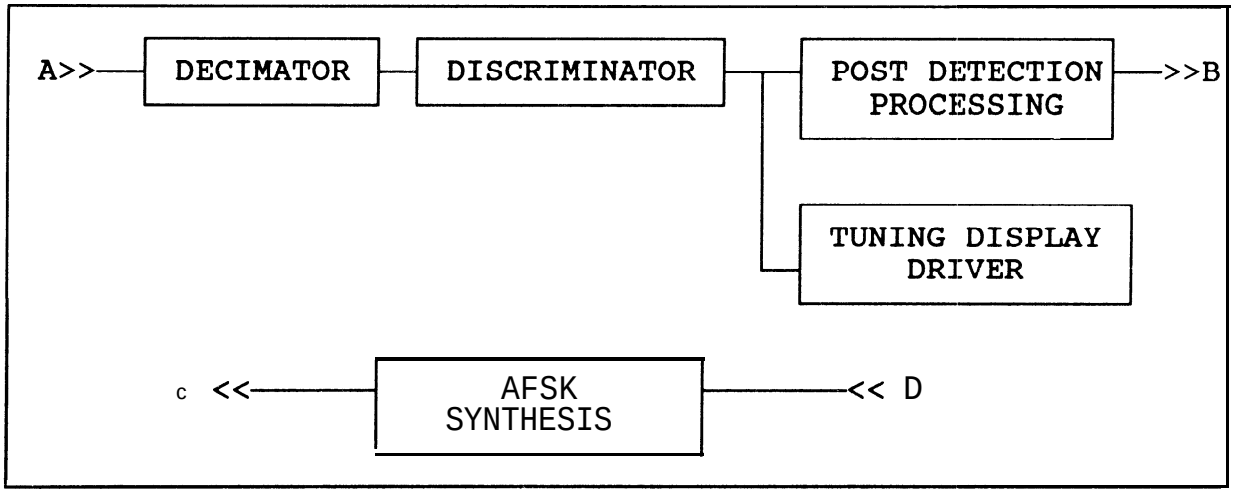


Figure 3. DSP modem architecture. The **analog** signal (A) is oversampled and decimated for increased dynamic range and better S/N. The discriminator consists of a pair of **matched** finite impulse response (FIR) filters. Post detection processing includes low pass filtering and thresholding for **data** output (B). A LED driver is provided for tuning purposes. Digital data for transmission is applied at (D) for digital audio frequency synthesis. This output, (C), is suitable for application to a SSB transceiver modulator.

DSP HARDWARE

Until fairly recently, DSP development tools, were very expensive and inaccessible to the average amateur experimenter. Recently, however, Texas Instruments (TI) released the TMS320C2x DSP starter Kit, also called the "DSK" (part number:TMDS3200026). For $99, the kit provides a small circuit module containing some DSP hardware, a thick user's guide, and a PC-based software package. All the user needs is a RS232 cable and 9V AC wall-mounted power transformer. This was intended as a low-cost introduction to DSP, and was received with great enthusiasm throughout the DSP community. The demand for the unit is so high that presently there is a world-wide shortage.

The software included a limited assembler, a nice debugger that executed the users' code on the DSP for real time debugging, and some coding examples to get you started. The DSP hardware module is a tiny circuit board measuring only 3.5 x 2.5 inches. All parts are surface-mounted which means that repairs would be very difficult. The DSP is a 40 MHz 320626 that has 1568 words on-chip static ram and a 256 word factory programmed ROM. At this clock rate, instructions typically execute in 100 ns. The on-chip RAM is configurable in several ways of code and data space configurations. The factory programmed ROM contains a simple bootstrap loader that allows the DSP memory to be loaded either from external memory, or via a RS232 line. The circuit module also contains a TLC32040 AIC that provides for a single channel, 14-bit A/D - D/A with sample rates as high as 44 kHz. The AIC is fully programmable and contains amongst-other things, programmable switched capacitor anti-aliasing and reconstruction filters.

Although the amount of RAM appears very limited, it actually goes a long way, in fact enough to implement our matched-filter HF demodulator and synthesized audio modulator.

DSP SOFTWARE

Sample Rate

As an example, software for a 2125 Hz mark, 2295 Hz space (170 Hz shift), 100 baud FSK demodulator will be shown. Specific details of the actual implementation of the demodulator should be read in conjunction with the source code. For availability of the DSP modem source code, please see the appendix.

The sample rate of matched filters for the DSP demodulator implementation is set at 6250 Hz. Since the space tone is set at 2295 Hz and the 100 baud modulation rate will extend the side

lobes of the spectrum to some extent. It thus appears that the chosen sample rate is adequate, (6250 > 2 x 2300 Hz).

Input Stage (decimator)

Consider the input decimator. The function of this stage, as previously discussed, is four fold:

a) Provide rejection of out-of-band signals, i.e. reject signals below 2125 Hz or above 2295.

b) Increase the dynamic range of the demodulator, i.e. for a 14-bit A/D the dynamic range is $20\log(1/(2^{14}-1)) = -84$ dB. It will be shown that the decimator used in this DSP demodulator behaves like a 21-element moving-average filter. The effect of such an arrangement is that each data value output is the result of a complex interpolation, i.e. each of the $2^{14}=16384$ quantized steps of the A/D convertor is further subdivided into much smaller steps. The actual dynamic range of such input stage will thus be in excess of 84 dB, which is quite impressive, however required for limiterless operation.

c) Increase the S/N ratio. If we assume that the input noise has a truly random behavior, the moving-average type input filter, will then by virtue of the additive nature of the signal component and noise components, cause the signal component to increase, while the noise component will tend to cancel. Unfortunately, some types of noise, such as static: crashes etc., does not behave in this way.

d) Reduce the sample rate for the matched-filter discriminator to.6250 Hz.

Decimators are actually rate downconvertors. They function by taking input samples, pushing them through a low-pass filter, and returns every N-th filter output for the result. The reason for the lowpass filter is to avoid new aliasing products being formed due to the lowered output sample rate. Since the matched-filter discriminator operates at 6250 Hz, the decimator is a x2 downconvertor and thus designed to operate at a sample rate of 12500 Hz. A special bandpass filter,, shown in Figure 4, is used with this decimator instead of the usual low pass filter,

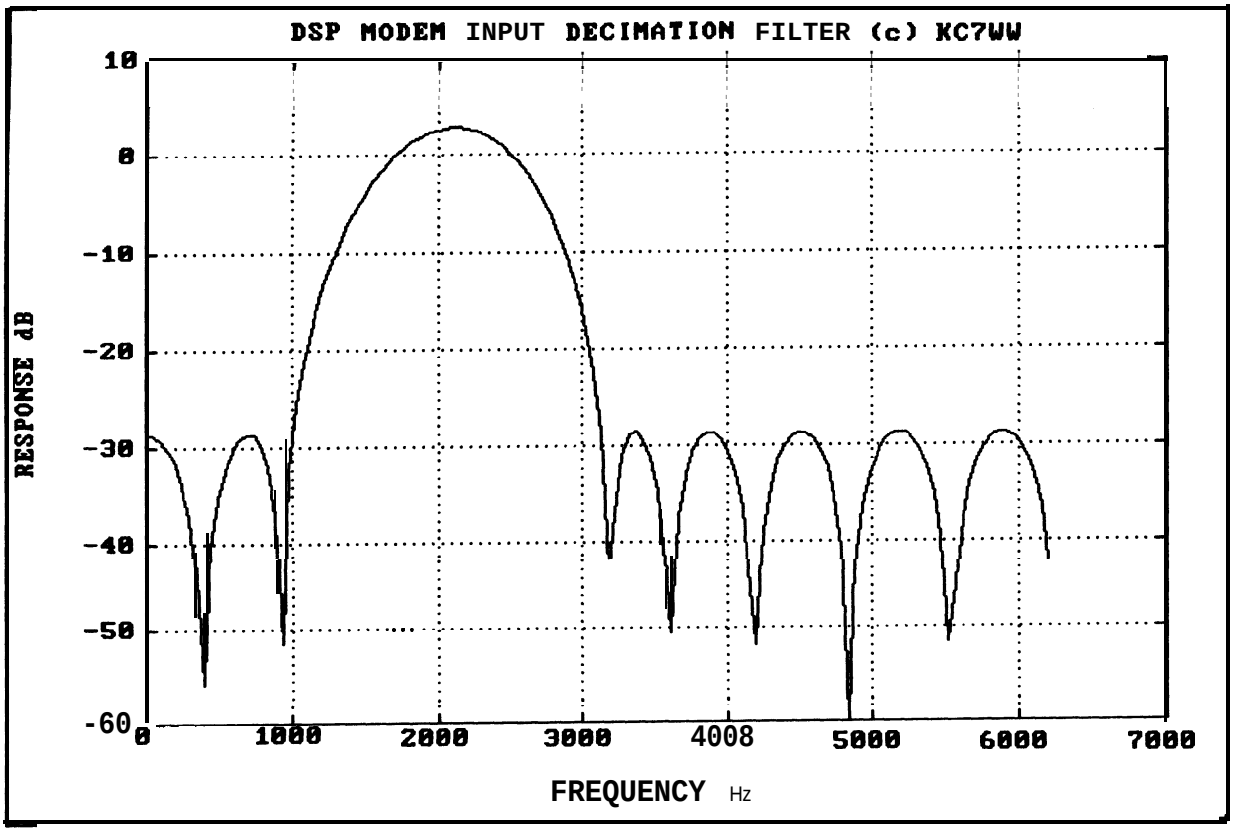
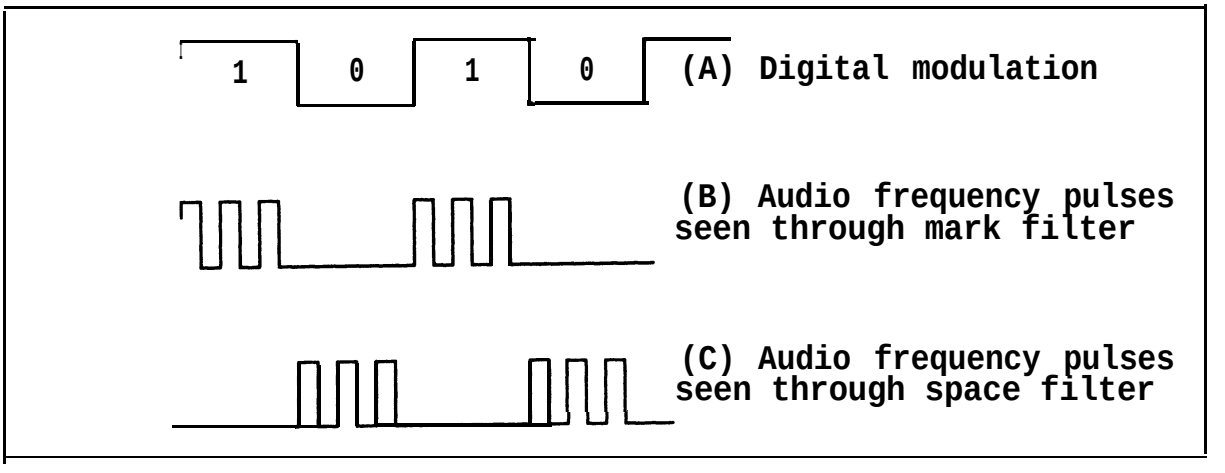


Figure 4. Input decimator filter. Note a bandpass filter is employed instead of the usual lowpass filter. Input sampling rate is 12500 Hz, output rate is 6250 Hz.



Fiqure 5. Summary of signals at various places in the discriminator. (A) The digital modulation, (B) mark, (C) space.

Matched-filter discriminator

A matched filter is simply the inverse of the impulse response of the wanted signal. Consider an input train of alternating zeroes and one's (Figure 5 (A)) where each bit time represents one signalling element, i.e. a baud. Then for 100 **baud rate,** each signalling element duration would be **1/100** th second. A signal period as seen by each filter, though would be twice this duration, or only 50 Hz as shown in Figure 5 (B) and (C). The matched filter should thus only respond to these tone pulses that has a repetition rate of 50 Hz. It can be shown that such an idealized filter would require very steep skirts, i.e. nearly infinitely small band-edge transition zones. Approximating such a filter in DSP would also require an inordinately large filter order, Practical use of the demodulator relies on the ability of **a** human operator to **"tune"** to received tones. This implies that the design should include some degree of tolerance, however, too loose tolerances will degrade performance. The DSP filters used in this DSP demodulator is shown in Figure 6.

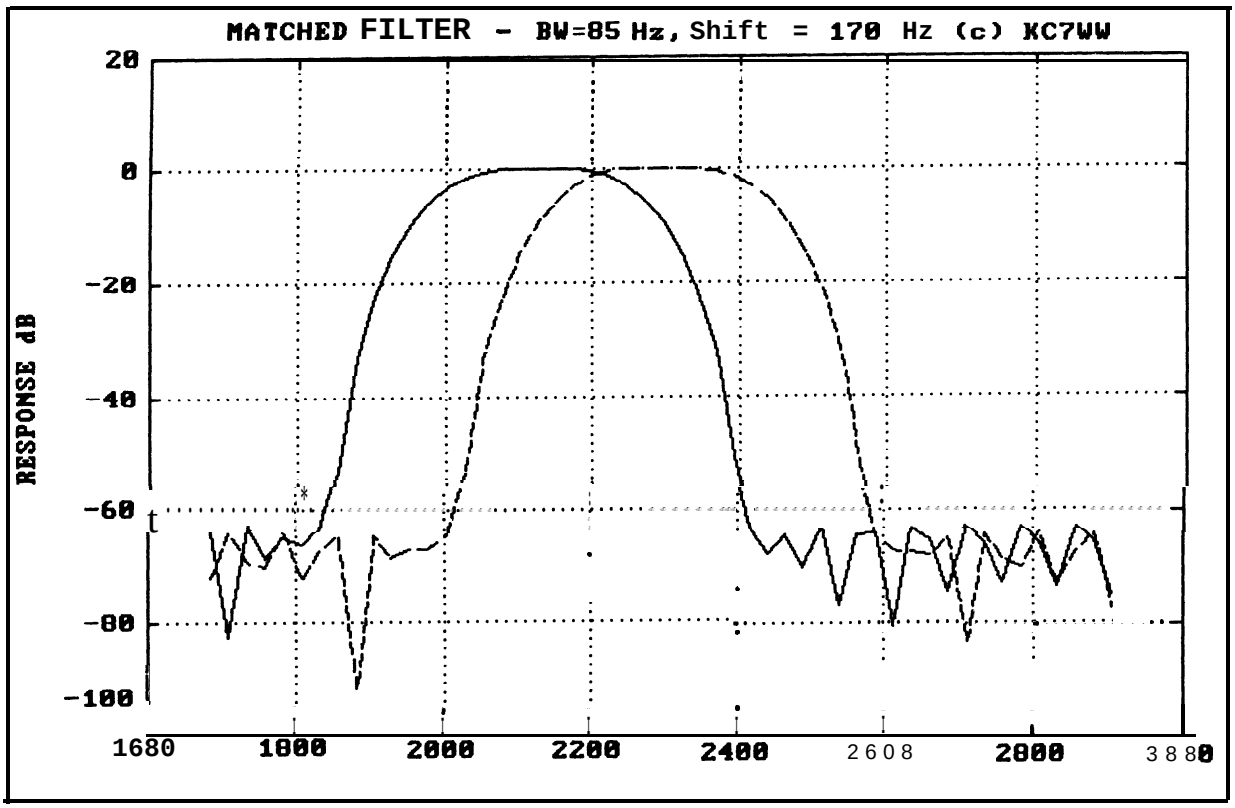


Figure 6. Matched filter pair as used for the DSP descriminator. This example shows 85 Hz wide, 81 th order FIR filters centered at 212512295 Hz. Sampling frequency is 6250 Hz. The passband of such filters are somewhat wider than an ideal matched filter to accommodate some degree of operator and equipment tolerances.

The discriminator filters as presented in Figure 6, shows minimum ripple in their passband, excellent rejection (better that 60 dB) and very steep skirts. The final function left in the discriminator, is envelope detection and signal combining. This is a rather simple task in DSP as all that is required is to take the absolute values of the discriminator signal filters and determine their difference. There still remains some audio frequency components in this difference signal, the removal of which is the subject of the next section.

Post-Uescriminator processing

The highest audio tone of interest is 2295 Hz, while the recovered modulation frequency is only 50 Hz. It is thus obvious that a simple low pass filter may be used to remove the unwanted

audio frequencies. If a cutoff frequency around 50 Hz is as well as sharp rolloff characteristics, then only the 50 Hz modulation will pass, however, the intended square-wave modulation signal will be degraded to a signal with heavily rounded edges. Some applications, such as RTTY, such slightly distorted waveforms is adequate as asynchronous character transmission is employed and it is relatively easy to estimate the center of each bit. In other instances, i.e. AMTOR and PacTOR, bit transitions are used for purposes such as bit phasing and the ability for accurately locating the bit transitions will influence the overall performance of such systems. For this reason and low pass filter with a gentle rolloff characteristic is more suitable.

There is a further consideration for the low pass filter design. At the sample rate of 6250, the output of the low pass filter would be updated every 160 microseconds. This translates to a small amount of uncertainty to where the exact bit transition occurs. As it turns out, a 6250 Hz sample rate DSP low pass filter with a cutoff frequency of 50 Hz, would require a relatively high filter order but if the sample rate could be reduced, a lower order filter with better characteristics could be designed. This DSP demodulator uses every other sample from the descriminator, i.e. reducing the sample rate to 3125 Hz before the actual low pass filter. This arrangement is effectively is also a decimator, and like in the case for the input decimator, has some desirable features that will improve overall performance. The low pass filter response is shown in Figure 7. Note that this is a very gentle low--pass filter that have been found to be adequate for most applications,i.e. RTTY, AMTOR, PacTOR, and HF Packet.

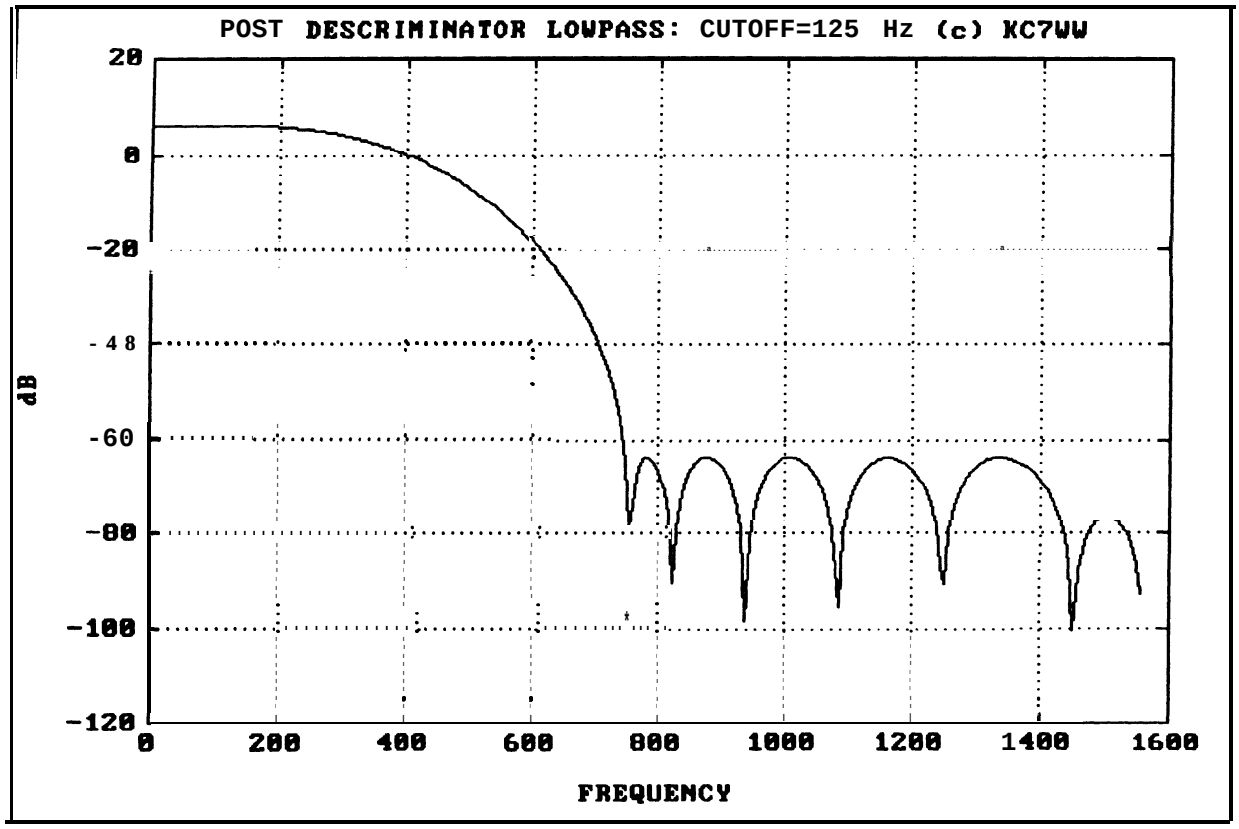


Figure 7. Post discriminator low pass filter. Cutoff, 150 Hz, filter order 15.

PERFORMANCE COMPARISONS

Performance testing and quantitive modem comparison is indeed a very difficult task. Engineering mathematicians often derive so-called "likelihood" (so-called BER) functions to estimate probable error rates in the presence of disturbances. However, this assumes that one can model the interaction of a multitude of variables such as modem architecture (whether it has a limiter or not) as well as the nature of noise on the different HF bands. This has proven to elude even the most basic of questions. What has been done successfully, however, is the application of specialized electronic atmospheric simulators. These "black boxes" is used to compare different demodulators under similar simulated "band" conditions. The DSP demodulator described in this article has not yet been tested using such sophisticated equipment, however, extensive testing against a high performance analog modem (please see AN-93 modem listed in bibliography). Under good conditions, no discernable difference could be found,

however under very adverse conditions the DSP demodulator has been found to be as good, perhaps marginally better.

SUMMARY AN CONCLUSIONS

A low cost DSP based modem for HF digital experiments have been described. It was shown that nearly ideal filters could be implemented in DSP with relative ease. Besides the added cost/performance benefits offered by a DSP approach, implementation of different modems is just a matter of downloading new code to the DSP. This flexibility implies that optimal modems for each application is readily available, something that is nearly impossible to achieve with an analog counterpart.

The author wishes to acknowledge that this article is based on the works of many gifted individuals without whose generous contributions this would not have been possible. A short bibliography is provided for further reference.

A source listing for the DSP modem, including a schematic for interfacing the DSK to an HF transceiver is available on the ADRS bulletin board for downloading as file HFDSP.ZIP

BIBLIOGRAPHY

(1) Goacher, D.J. (G3LLZ) and J.G. Denny (G3NNT) 1.973. The Teleprinter Handbook. RSGB, 35 Doughty street, London WC1N 2AE.

(2) Petersen, K. (W8SDC), I. Hoff (K8DKC), V. Poor (K3NIO) 1964. The Mainline TT/L. RTTY Journal (November).

(3) Petersen, K. (W8SDC), I. Hoff (K8DKC), V. Poor (K3NIO) 1967. The Mainline TT/L-2. RTTY Journal.

(4) Hoff, I. 1970. Mainline Solid State Demodulators. RTTY Journal, September, October, November.

(5) Pietsch, H-J. (DJ6HP) 1975. Der RTTY-Nf-Konverter DJ6HP 001. CQ-DL 12/75.

(6) Walker, J.D. (G6FYU) 1985. Multistandard Terminal Unit. Electronics and Wireless World.

(7) Petit, R. (W7GHM) 1975. Coherent CW - Amateur Radio% New State of the Art? QST September.

(8) Petit, R. (W7GHM) 1991. CLOVER-II: A Technical Overview. ARRL

Amateur Radio 10th Computer Networking Conference Proceedings.

(9) Osborne, P.W. and M.B. Luntz. 1973. Coherent and Noncoherent detection of CPFSK. IEEE Transactions on Communications. VOL. COM-22 no. 8.

(10) Baghdady, E.J. 1955. Frequency-modulation interference rejection with narrow-band limiters. Proceedings of the IRE. January 1955, p51-61.

(11) Baghdady, E.J. 1958. Theory of stronger-signal capture in FM reception. Proceedings of the IRE. April 1958.

(12) Hamming, R.W. 1983. Digital filters. Prentice Hall

(13) Rorabaugh, C.B. 1990. Communications formulas & algorithms for systems analysis and design.

(14) Parks, T.W. and C.S. Burrus. 1987. Digital filter design. Wiley.

(15) Proakis, J.G., C.M. Rader, F. Ling, and C.L. Nikias. 1992. Advanced digital signal processing. Macmillan

(16) Reyer, S.E. and D.L. Hershberger. 1992. Using the LMS algorithm for QRM and QRN reduction. QEX (127) September 1992.

(17) Hershberger, D.L. 1992. Low-cost digital signal processor for the radio amateur. QST September 1992.

(18) Forrer, J.B. 1994. AN-93, and HF Modem for RTTY, AMTOR and PACTOR Software TNC's. QEX, May.

(19) Texas Instruments. 1986. Digital signal processing applications with the TMS320 family. Volumes 1 - 3.

(20) Park, S. Principles of sigma-delta modulation for analog-to-digital convertors. Motorola Digital Signal Processors Application note APR8/D.

(21) Taylor, F.J. 1983. Digital filter design handbook. Dekker.

(22) Oppenheim, A.V. and R.W. Shafer. Discrete-time signal processing. Prentice Hall.

(23) McClellan, J.H., T.W. Parks, and L.R. Rabiner. 1973. A computer program for designing optimum FIR linear phase digital filters. IEEE Transactions on audio electroacoustics. Vol. AU-21(6). p506-526.

G-TOR™: The Protocol

by Kantronics Staff
Mike Huslig, Phil Anderson, Karl Medcalf, and Glenn Prescott

Foreword

The G-TOR data communications protocol is an innovation of the technical staff of Kantronics Co., Inc. It was introduced as an inexpensive means of improving digital communications in the HF radio bands. A hybrid ARQ scheme, used in combination with an invertible, half-rate Golay forward error correcting code, is the single-most essential element in the protocol.

The purpose of this document is to present a detailed description of the G-TOR protocol. It is assumed that the reader is familiar with ARQ systems such as AMTOR, Pactor, and Packet; terms such as MASTER, SLAVE, ISS, IRS as they pertain to protocols; and binary, HEX and C-language number notations. Operation, performance objectives, and performance results of systems using this protocol are not discussed; these aspects of G-TOR have been covered widely in trade publications.

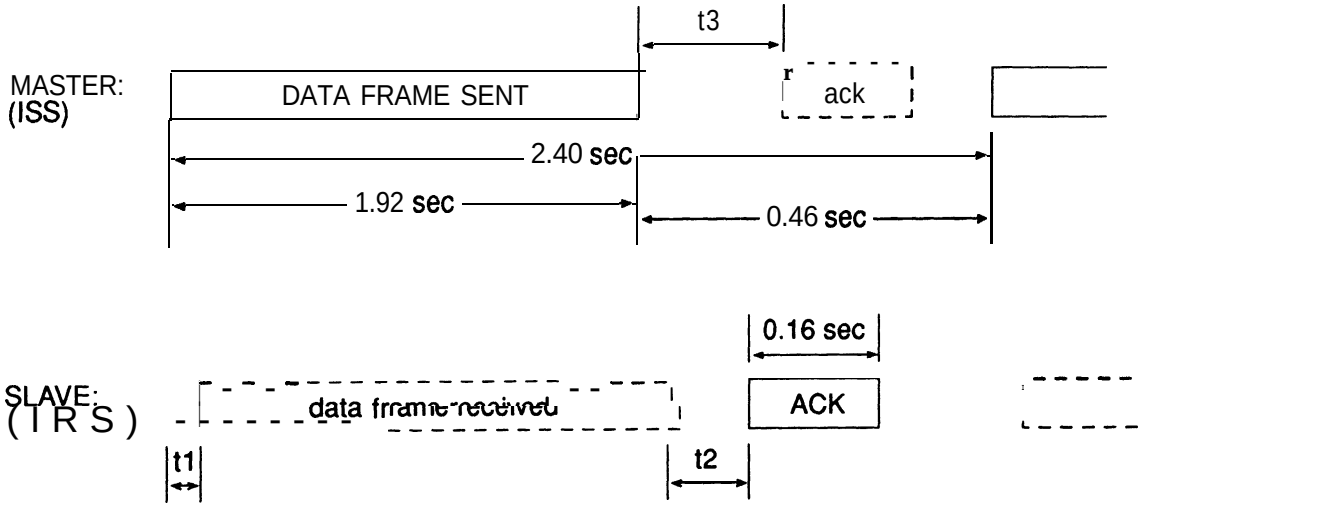
The description is organized in sections as follows: a general overview, including term definitions and initialization of parameters; timing; definition and usage of data, control, BK, and connect and disconnect frames; data formats; speed change procedures; the Huffman table; and Golay coding and data interleaving. Appendices containing flow charts, a Huffman decoding tree, and a C language routine for Golay encoding/decoding follow the protocol description.

General

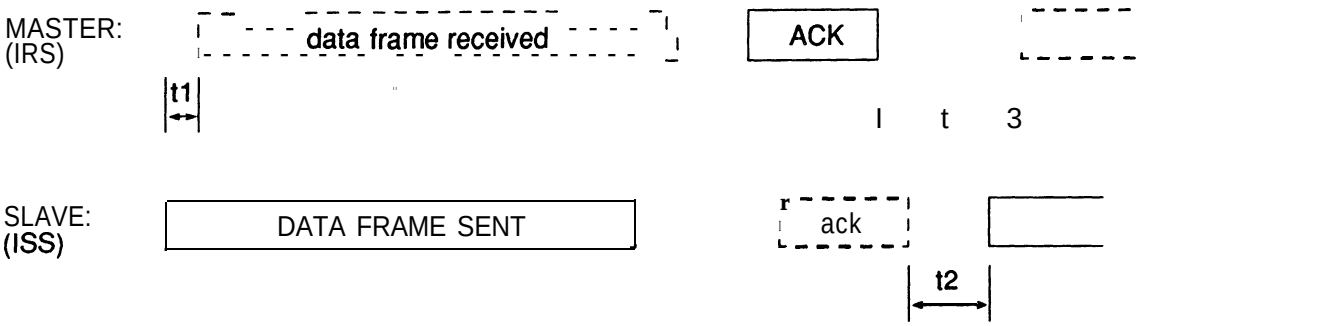
The system which originally transmits a G-TOR connect request is called the Master, and the system which responds to the transmitted connect request is called the Slave. The system currently sending data blocks is called the Information Sending Station (ISS), and the system receiving these data blocks is called the Information Receiving Station (IRS). During a connection, tlhe Master is always the Master and the Slave is always a slave, but either system may be the ISS while the other is the IRS. Immediately after a connection, the Master is the ISS while the Slave is the IRS. The IRS will send a Control Signal #1 (CS1) immediately after connection or turnaround to indicate it is ready for data; it sets an internal flag (Send_CS_flag) to CS1. The IRS also sets its internal error count to 0, its blocks_received count to 0, and its Last Block number to 0. When the ISS receives the CS1, it sets an internal flag (Expecting_CS_flag) to CS2 and Block_number to 1.

The Master and the Slave both have an internal flag (Golay_flag) which is complemented every 2.4 second cycle. During the connect process, this flag is set to be the same in both systems. Whenever the ISS receives a proper Acknowledgment (the CS1 the first time around), it forms a new frame of data (Real_Data). This new data frame is also fed through the Golay encoder to form a frame of parity bits (Golay_Data). The ISS sets an internal error count to 0. Depending on the state of the Golay_flag in the ISS, the ISS will choose the Real_Data frame or the Golay_Data frame to transmit. The Golay_flag in the ISS is then complemented for the next cycle. Whichever frame is chosen, that frame is then interleaved and transmitted.

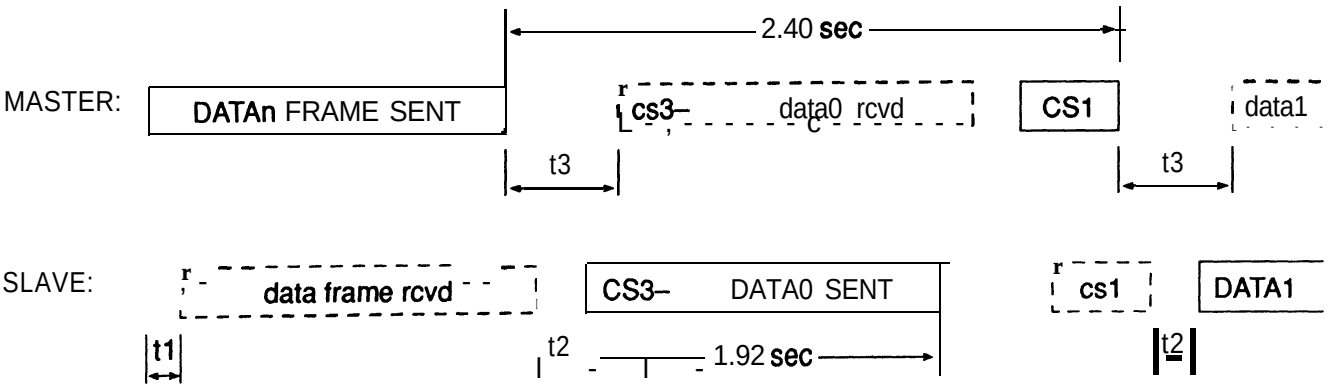
Figure 1



tl is radio wave propagation time
t2 is slave acknowledgment delay which includes processing time and transmitter turn-on delay; t2 is constant while connected, even when the Slave is the ISS
t3 is determined by the Master during initial synchronization and should vary only slightly during the connection



Changeover timing



The IRS is expecting to receive a frame during a certain time period in the 2.4 second cycle. When it has received the frame, the IRS then increments its Blocks_received count and de-interleaves the block. If the ISS Golay_flag is set, a copy of the block is saved as Golay_Data; the block is then fed through the Golay encoder to generate the original data. If the ISS Golay_flag is clear, a copy of the block is saved as Real_Data. The Golay_flag is then complemented. If the CRC of the block is correct, the IRS has received the data correctly. If the CRC is not correct, the IRS checks to see if the Blocks_received counter is greater than 1, indicating it has received a copy of both the Real_Data and the Golay_Data. If the IRS has a copy of both, it will try to regenerate the original data using Golay error correction. If the CRC is still incorrect, the IRS error count is incremented. If the error count is greater than a set maximum number of errors, the IRS will go back into a standby mode; otherwise, to indicate failure, the IRS will re-send the same Control Signal it sent in the last cycle. If the CRC of the received block or the error corrected block is correct, the IRS clears its Blocks_received count and compares the Block_number in the received frame with the Last_Block number it correctly received. If they are the same, then the received data frame is the same, indicating most likely that the ISS has not correctly received the last Control Signal sent by the IRS; the IRS then re-sends the last CS. If the Block_number is one greater than the Last_Block number received, then this block is the next data expected; the IRS now sets its Last_Block number to the Block number received and prints the data received; the IRS error count is set to 0; the Send_CS_flag is complemented and the appropriate Control Signal is transmitted. If the Block number is otherwise, then some protocol error has occurred, and data has been lost.

The ISS is expecting to receive an acknowledgment during a certain time period in the 2.4 second cycle. If the ISS receives a CS2 when it was expecting a CS2, or it receives a CS1 when it was expecting a CS1, the ISS considers the sent data to be properly acknowledged. The Expecting_CS_flag is complemented, the Block_number is incremented, and the ISS fetches new data to be transmitted. Otherwise, the data has not been acknowledged, or the ISS has not received the acknowledgment. The ISS then increments its error count, and if the error count is less than some set maximum, the ISS will try to send the data again.

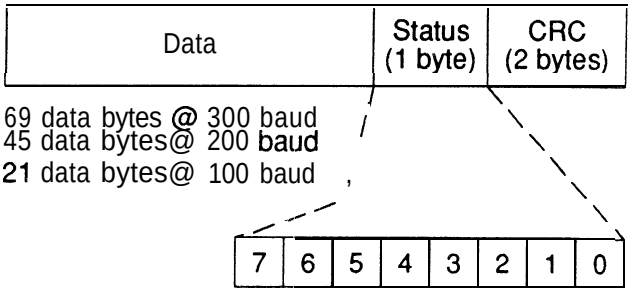
Timing

The basic G-TOR cycle is very similar to AMTOR and PACTOR. The ISS sends long data frames which are acknowledged by the IRS with shorter control signals (CS). The total cycle duration is 2.4 seconds. The data frames are 1.92 seconds long and the control signals are 0.16 seconds long. 0.32 seconds remain in the cycle for radio switching, wave propagation, and the necessary computing for both Master and Slave systems. The Master controls tlhe total cycle time. The Slave adjusts its receive window to follow the Master's transmissions, but since the Slave's transmissions are always fixed in relation to its receive window, the Slave's transmissions follow the Master's transmissions. The Master only corrects its receive window. Refer to Figure 1.

Data Frame Structure

The frame structure for a typical G-TOR data frame (before interleaving) is shown in Figure 2. The data frame is 1.92 seconds in duration. Depending on channel conditions, data can be sent at 100,200, or 300 baud. Each data frame is composed of either 72 bytes (at 300 baud), 48 bytes (at 200 baud), or 24 bytes (at 100 baud).

Figure 2
G-TOR Frame Structure Before Interleaving



A single byte near the end of the frame is devoted to command and status functions. The status byte is interpreted as follows:

- **status bits 7 & 6:**
  Command
  00 – data
  01 – change-over request
  10 – disconnect
  11 – connect
- status bits 5 & 4:
  Unused
  00 – reserved
- **status bits 3 & 2:**
  Compression
  00 – none
  01 – Huffman
  10 – Swapped **Huffman**
  ll-reserved
- status bits l&O:
  Block Number modulo 4

The last 2 bytes of the frame contain the CRC. Like Packet and Pact-or, the CRC is computed using the same CCITT standard, starting at the first byte of a data, connect, or disconnect frame and starting at the third byte of the BK frame. However, the two bytes of the CRC are swapped before being put in the frame.

Control Signal Structure

The G-TOR Control Signals (CS) are 2 bytes (16 bits) long and are always sent at 100 baud. Each byte of the Control Signal is sent LSB first. Control Signals are used to acknowledge correct or incorrect receipt of frames from the information sending station. They are also used to request changes in transmission speed and to initiate a change-over in information flow direction. There are five different G-TOR Control Signals:

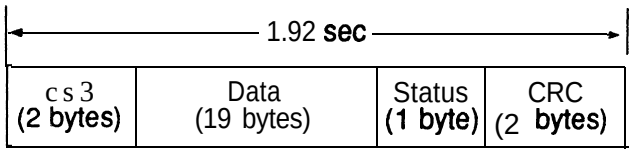
Signal-Function	Code	Bit pattern in time
CS1-Data ack/nack	**F11A**	1000111101011000
CS2-Data ack/nack	**6B62**	1101011001000110
CS3-Change-over command	**5E13**	0111101011001000
CS4-Speed change	**4D3C**	1011001000111100
CS5-Speed change	8957	1001000111101010

The CS codes are composed of multiple cyclic shifts of a single **15-bit** pseudo-random noise (PN) sequence (an extra '0' bit is appended to the sequence for balance, so the total CS word length is 16). A pseudo-random noise sequence is used because PN sequences have **powerful** mathematical correlation and distance properties which facilitate the identification of the appropriate CS code, even in the presence of noise and interference. Each CS has a mutual Hamming distance of 8.

BK Frame Structure

The change-over frame is shown in Figure 3. This frame is always transmitted at 100 baud and is never interleaved. It is essentially a combination of the CS3 Control Signal and a shortened data frame. Each byte of the BK frame is sent LSB **first.**

Figure 3
G-TOR Changeover Frame Structure



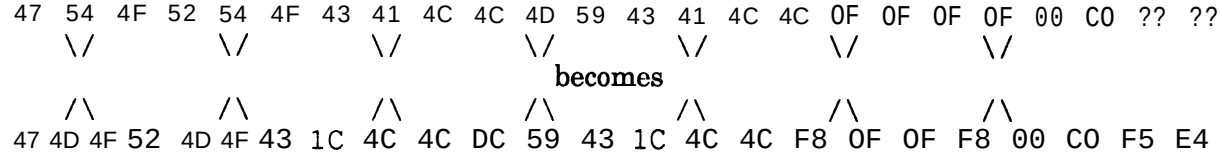
Formation of Connect and Disconnect Frames

Connect and Disconnect frames are always sent at 100 baud (24 bytes). The first 10 bytes contain the call/address of the destination and the second 10 bytes contain the call/address of the source. These 20 bytes use 7 bit ASCII. If the call/addresses are less than 10 bytes long, the fill character **0x0F** should be used to extend the addresses to 10 bytes.

The 21st byte is zero. Bytes 23 and 24 are the CRC. Byte 22 is the status byte and will be 11000000 for a connect frame or **100000xx** for a disconnect frame. Note that the Block Number for a connect frame is always 0.

The MSB of the first 20 bytes are originally zero because of the use of 7 bit ASCII. Bytes 2,

Figure 4



5, 8, 11, 14, 17, and 20 should now have their MSB set to one; then the nibbles of these bytes should be swapped. For example, a connect frame to GTORTOCALL from MYCALL would form as shown in Figure 4.

The reason for this strange format is that when the frame is broken up into 12 bit tribbles and sent to the interleaver,

474 D4F 524 D4F 431 C4C 4CD C59
431 C4C 4CF 80F 0FF 800 C0F 5E4

the first 14 bits transmitted (the MSBs of the tribbles) will be alternating 0s and 1s. Note that this pattern is not present when the Golay form of the frame is being sent.

The Slave should also look for connect frames with mark and space inverted, and the Master should also look for inverted Control Signals. Once connected, each station should remember its received polarity.

When the Slave decodes a connect frame addressed to it, the Slave would normally answer with a CS1 control signal. If the Slave is busy, it would answer with a CS2. If the 21st byte is not zero, or the 6 lower bits of the status byte are not zero, the Slave should answer with a CS5; this is for future expansion – the Master indicating it has added capabilities, the Slave indicating it does not yet support those capabilities.

The Slave must be careful about 'when' it acks the Master. Like Amtor and Pactor, the Slave sets a fixed time after the Master's transmission for its own transmission. For maximum propagation, the Slave should set this time as short as possible. However, the time should be long enough so that it can not only decode and possibly correct a data frame before sending the ack as an IRS, but also long enough to form a data frame when an ack is received from the ISS. In other words, the Slave must be aware of the time needed for its own processing.

The connect and disconnect frames are always sent at 100 baud. If the ISS wants to disconnect but is transmitting at a higher baud rate, it should send an idle frame with a status byte 100000xx; when the IRS sees this frame, it should send a CS5 to downspeed the ISS but should stay connected until the ISS sends a true disconnect frame.

After the IRS acknowledges a disconnect frame, it should remember the time relationship between the disconnect frame and the IRSs ack. If the ISS did not copy the ack, it will keep sending disconnect frames until it times out. If the IRS copies a disconnect frame to it while in standby, it should re-send the last ack.

Data Format in Frames

The ISS can send data in three forms: straight ASCII, Huffman compressed, and swapped Huffman compressed. Swapped Huffman uses the same tables as Huffman compressed but swaps the upper case letters with the lower case. Since Huffman compressed favors lower case letters as in normal text, Swapped Huffman favors upper case letters in text which may be predominately upper case. The ISS must decide in which form to send the data in order to provide the greatest throughput; if there is no advantage in sending Huffman codes, the ISS should send in straight ASCII. All normal data frames and connect and disconnect frames are interleaved and, on alternate cycles, Golay encoded.

If there is not enough data to send in a data frame, IDLE codes are used to fill the frame. If the frame is sending straight ASCII, 0x1E is used as the IDLE code. In order to send a 0x1E data byte, a 0x1C pass code must be sent followed by 0x7E; in order to send a 0x1C data character, a 0x1C pass code must be sent followed by 0x7C. Only the ASCII data characters 0x1C and 0x1E need a pass code. The pass code should never be the last character in an

ASCII data frame; in other words, the combinations 0x1C 0x7E and 0x1C 0x7C should never be split between data frames. G-TOR Huffman compression uses a unique IDLE code; there is no pass code when sending a Huffman compressed frame.

The IDLE code also indicates the end of data in a data frame: straight ASCII or Huffman compressed. The IRS should stop decoding the data frame when it encounters an IDLE code, and the ISS should never send data after an IDLE code in a data frame. This function is reserved for possible expansion.

BK Frames

If the IRS wants to send data to the ISS, it can seize the link and become the ISS by sending a BK frame. The BK frame is a special data frame which is always sent at 100 baud and is never interleaved nor Golay encoded. The first 16 bits of the BK frame comprise the CS3 control signal. The next 22 bytes are 19 bytes of data plus the status byte and 2 byte CRC formed over the data starting after the CS3. The Block Number in the status byte of the BK frame is always 0. Each byte is sent LSB first. If the ISS receives the BK frame correctly, it sends a CS1 and becomes the new IRS, expecting new data frames at the previous baud rate. If the ISS detects the CS3 but does not receive the data correctly, it sends a CS2 and becomes the new IRS, still expecting data at the previous baud rate. If the original sender of the BK frame received a CS2, it will re-form a data frame using the old data that was used in the BK frame plus any additional data available, but again at the baud rate in use before the BK frame was sent. If the sender of the BK frame receives neither a CS1, CS2, or CS3, it will re-send the original BK frame.

Since there is a possibility that the ISS does not receive the CS3 part of the BK frame and therefore will re-send a data frame or the Golay encoded form of the data frame, the ISS must ensure that any data frame or Golay encoded form of a data frame will not produce a waveform which would appear as a 100 baud CS1, CS2, or CS3 in the time slot where the IRS may be looking for an acknowledgment to its BK frame. The IRS should be sampling in the receive ack time slot at the previous baud rate to ensure that the ack received is truly a 100 baud signal and not an artifact of the ISS data frame at a higher speed.

The ISS can request a changeover by sending a data frame with bit 6 of the status byte (BK request bit) set to 1 (0100xxxx); the IRS would then send a BK frame. A BK frame can also be acknowledged with another BK frame, causing quick changeovers. The BK frame serves as a positive acknowledgment of the previously received data.

Changing Speed

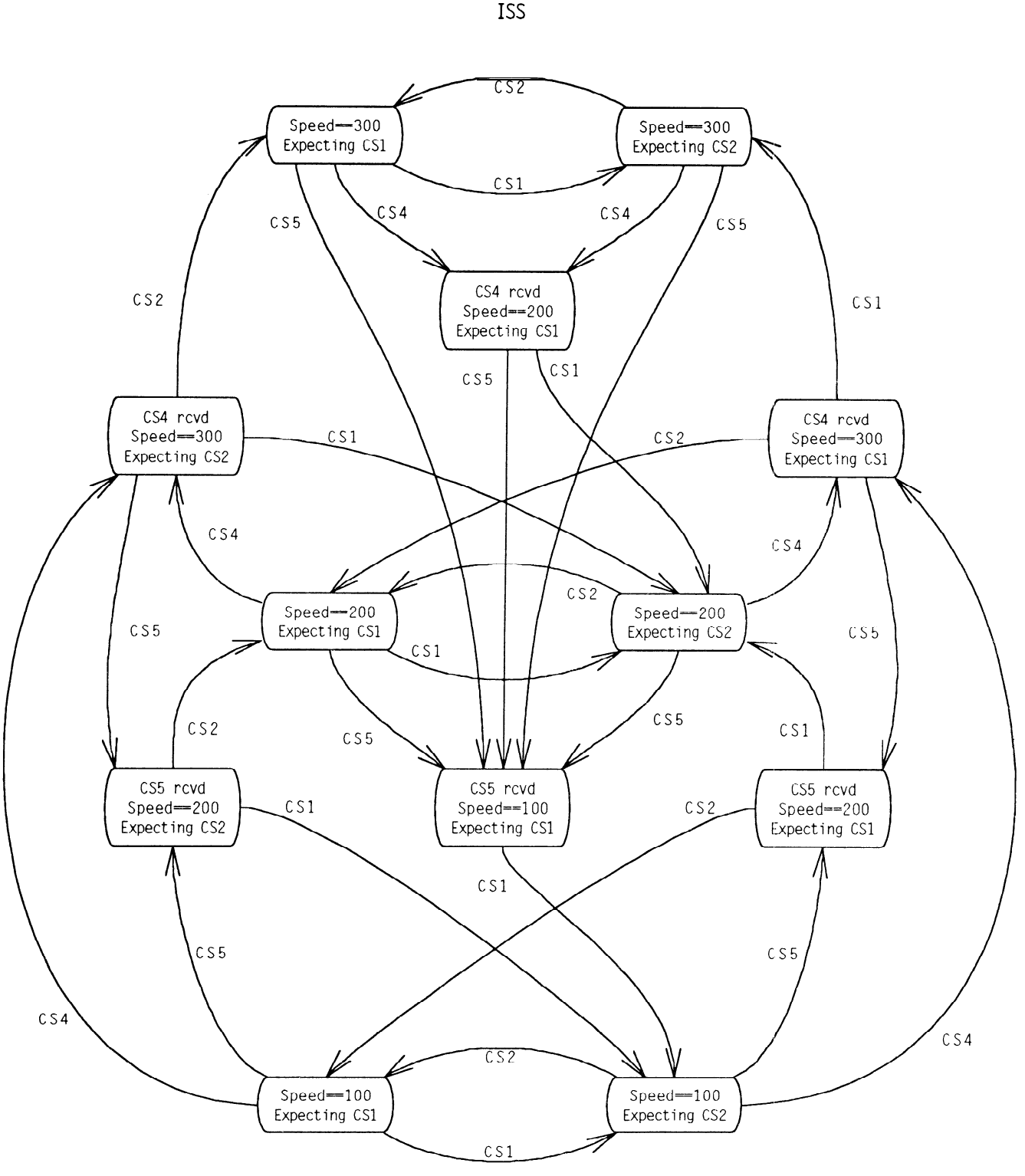
Data frames can be sent at 100,200, or 300 baud. CS4 and CS5 are the Control Signals that the IRS uses to change the sending speed of the ISS. Since the IRS can cause the ISS to change from any one speed to any other speed, the Control Signal used by the IRS depends on the states of the two systems. Refer to the Speed Transition Diagram in Figure 5. The algorithm used by the IRS to determine speed changes is not a part of this protocol. The algorithm used by the KAM, however, is shown in the flowcharts. A speed-up CS always acts as a positive acknowledgment of the previous data frame. A speed-down CS asks for the previous data to be re-sent at a slower speed.

Slowdowns and BK Frames

If the ISS receives a slowdown signal from the IRS, it has no way of knowing whether the data just sent was received correctly or not and therefore should re-send the data at the requested slower speed using the same block number. It is possible that the IRS could request a further slowdown in speed while the ISS is re-sending data. Any time the IRS receives valid data, it should keep a count of the characters in the frame. If the IRS slows the ISS down and the new data frame received has the same block count as the previous frame, the IRS knows the ISS is resending data and should throw away the appropriate number of characters. The ISS and IRS need to be careful with these character counts during double slowdowns (from 300 to 200 and then from 200 to 100 baud).

If the IRS tells the ISS to slow down after the ISS has sent a data frame with the BK request bit set, or if the ISS decides it wants to send a

Speed transition diagram
Figure5



BK request while re-sending data in response to a slowdown, the ISS should not set the BK request bit in the slower data frames until the data frame contains the last character sent in the original.

The IRS cannot send a BK frame until it receives a valid data frame since the CS3 of the BK frame is an acknowledgment of valid data. If the IRS is receiving duplicated data due to a slowdown, it should not send a BK frame until all the duplicated data is received.

RLEn Coding

An RLEn code is a 19 bit code made up of a unique 14 bit Huffman code followed by 5 bits which represent a number n, O-31. RLEn codes are found only in Huffman compressed data frames and can never be the first code in a data frame.

When an RLEn code is encountered in a data frame, the previous character decoded in the frame should be repeated an additional N times, where N is a number which depends on n and the number of bits used by the previous Huffman character according to the following table.

length of previous character	N
2 bits	n+10
3 bits	n+7
4 bits	n+5
5-6 bits	n+4
7-9 bits	n+3
10-16 bits	n+2

An RLEn code may follow another RLEn code immediately, indicating that the previous code, which was just repeated, should be repeated an additional N times.

Huffman codes are put into the data fields in the order shown in Appendix 11. For example, the first few bytes of "The quick brown fox" using Huffman compression would be formed as shown in Figure 6.

And before interleaving or Golay encoding, the bytes are grouped into tribbles

1A2 3BD 6FE A65 . . . .

Golay Coding and Interleaving

Before a data frame is transmitted, the data is regrouped into 12 bit tribbles. For example, a 100 baud frame of "The quick brown fox" using no compression would be formed like:

```
54 68 65 20 71 75 69 63 6B
20 62 72 6F 77 6E 20 66 6F
78 1E 1E 01 7E 64
```

And then grouped into tribbles

```
546 865 207 175  696 36B
206 272 6F7 76E 206 66F
781 E1E  017 E64.
```

The data is interleaved by sending in time the MSB of each tribble, and then the next MSB, etc. The bit sequence of the above data would start:

```
  time->
0100000000000101
1000100011011101
0010111111111101
1001010001001000
... 8 more groups of 12 bits
```

The ISS alternately sends frames of data and Golay encoded data. Golay codes are unique 1%bit codes derived from 12 bits of data. The C program in Appendix 10 shows how to generate the codes from the data and also how to regenerate the correct data from the 24 bits of data and Golay codes which have errors. The correction algorithm will correct up to 3 bits in error from the 24 bits of data and encoded data. The Golay codes are generated from the

Figure 6

```
      T       h    e          q        u    i     c       k
0001101  000100 011  10 1111010110 11111 1101  010011  0010101
00011010  00100011  10111101 01101111 11101010  01100101 01......
   1A       23       BD       6F       EA       65
```

tribbles of data before interleaving, so that "The quick brown fox"

```
546 865 207 175 696 36B 206 272
6F7 76E 206 66F 781 E1E 017 E64
```

becomes

```
083 092 57B 1A7 F88 C46 A85 AF1
9AE 342 A85 291 114 BAF 0B1 3F0.
```

The tribbles are then interleaved as before, starting with the MSB of the first tribble.

Note that the CRC of the original data is also Golay encoded; there is no CRC generated over the Golay encoded frame.

Note also that the inverse Golay function is identical to the Golay function; in other words, $x=g(g(x))$.

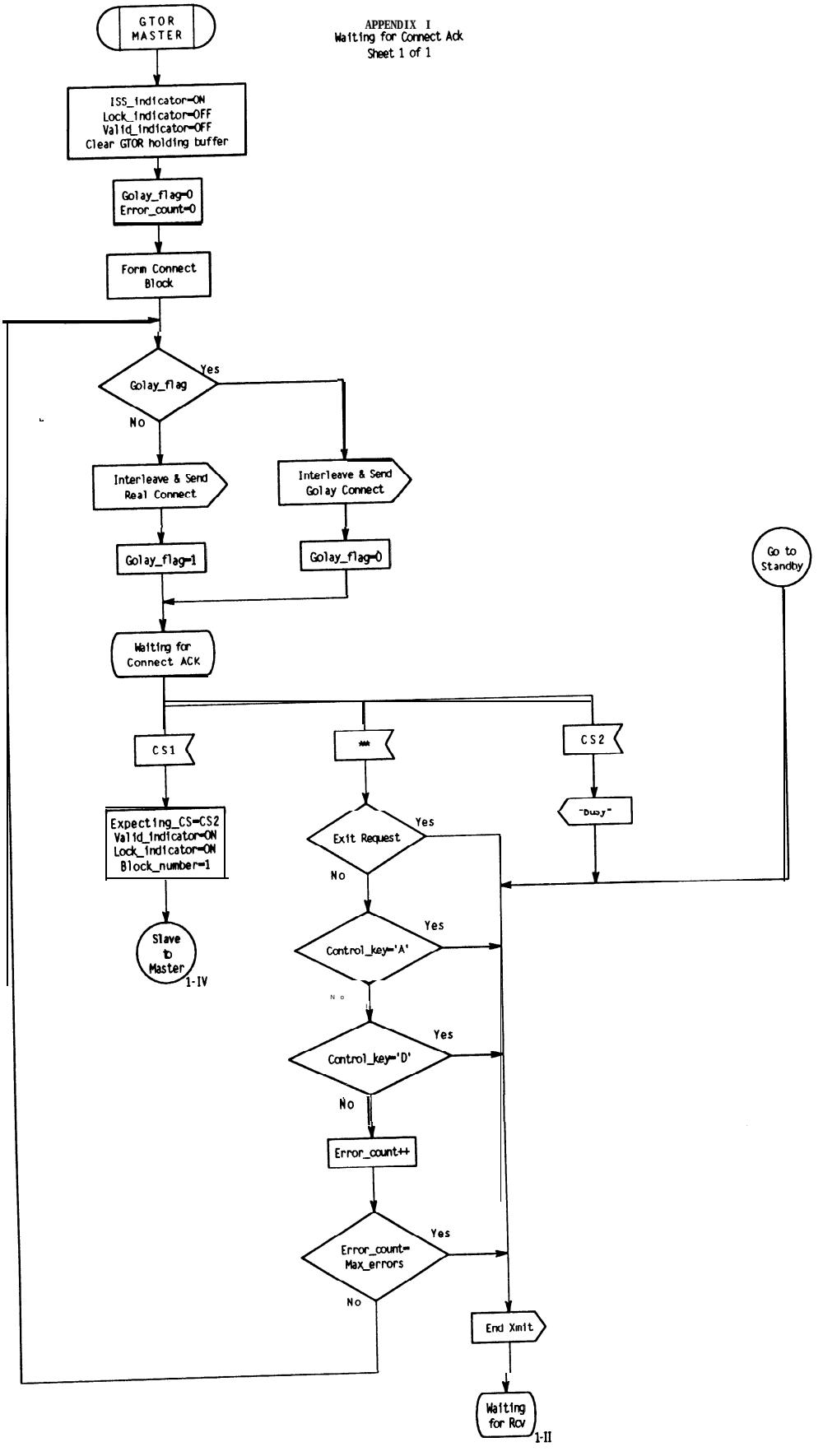
FEC Transmissions

At this time there is no special G-TOR broadcasting mode. AMTOR mode B is used for calling CQ. A G-TOR unit in standby should be able to receive AMTOR mode B FEC signals.

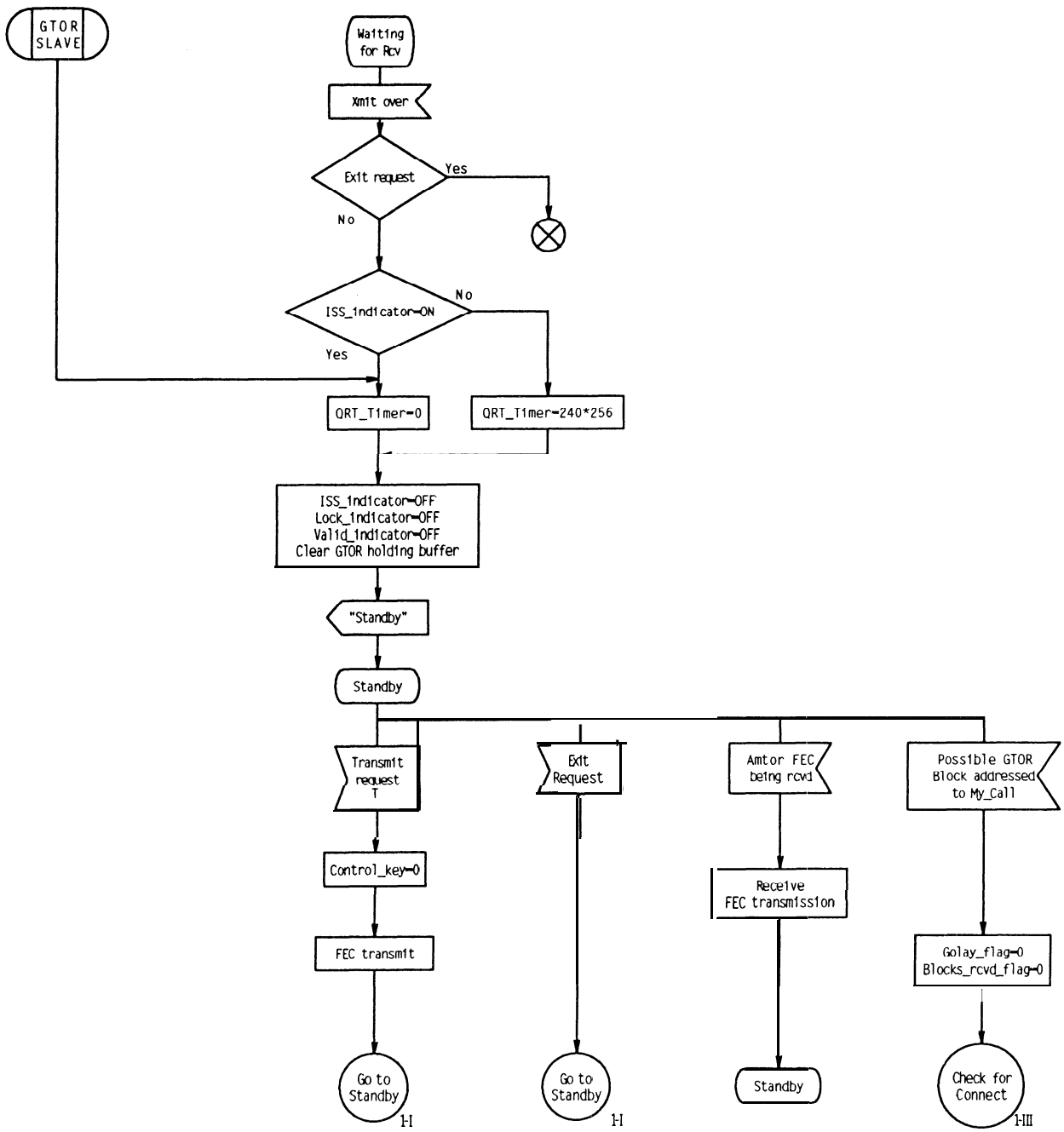
Monitoring G-TOR

Third party monitoring of G-TOR connects can be very difficult due to the nature of the G-TOR protocol. Although a data frame is always 1.92 seconds long, it may have been sent at 100,200, or 300 baud. The frame received may be the Golay encoded form of a data frame. The BK frame is different in that it is not interleaved, and its CRC is calculated over shortened data. The frame received could also be inverted polarity; however, the inversion would stay the same during any particular connection. Since the Golay error correction allows the IRS to copy data without ever getting a proper CRC in one frame, a monitor program should also go back 2.4 seconds to form a correct frame if it is to be thorough. Again, because of the nature of the G-TOR signal, Carrier Detect or a PLL on bit transitions cannot be used reliably, but a brute force algorithm can be used. It would sample the data stream at twice the baud rate for 100,200, and 300 baud. Sampling at twice the baud rate will take care of problems caused by sampling near the edge of a bit. A program was written to do a brute force algorithm using the fastest assembly language techniques to check for all possible G-TOR frames; the program ended up using about 1/3 of the available cycles of a 50 MHz 486DX.

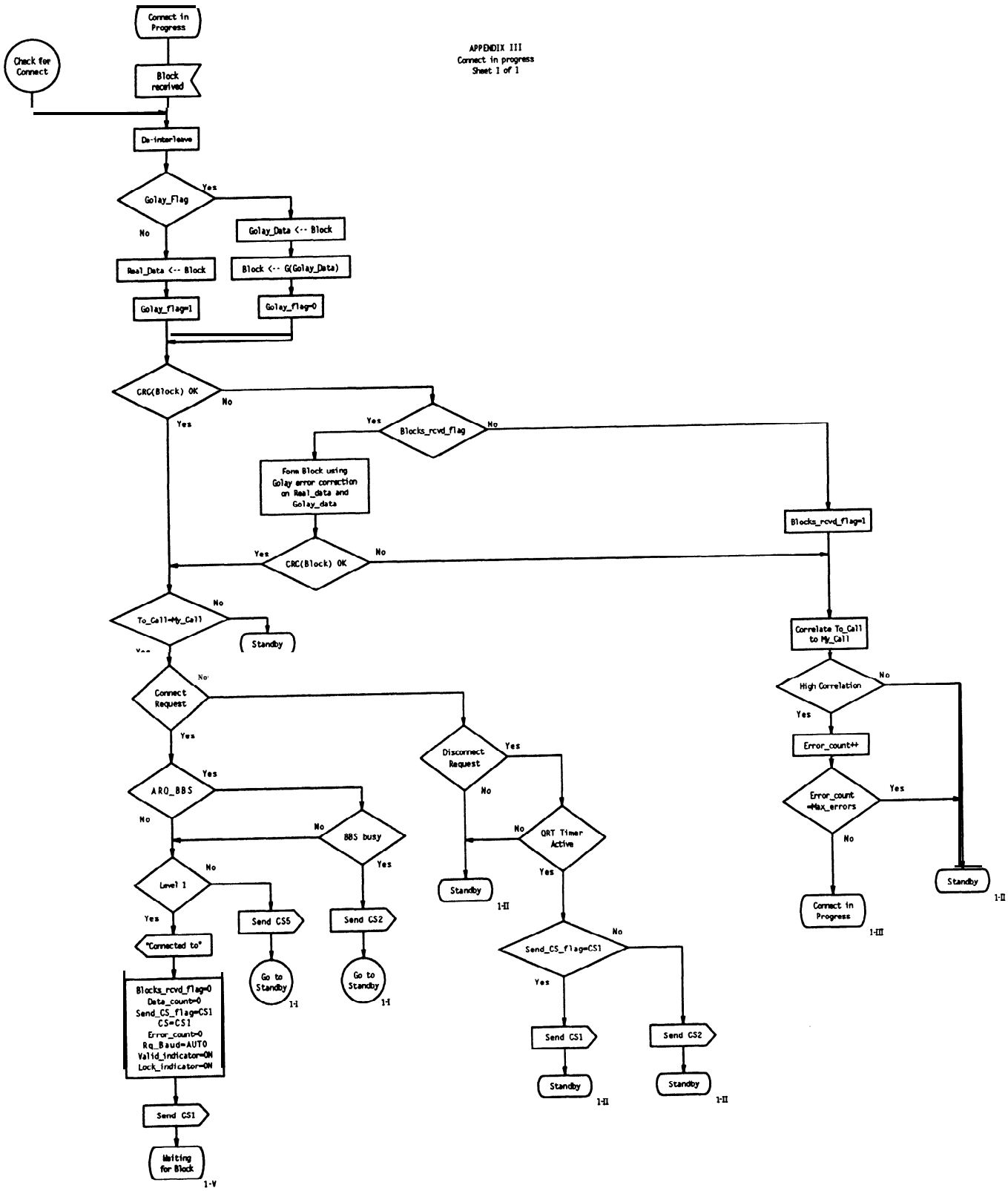
APPENDIX I
Waiting for Connect Ack
Sheet 1 of 1

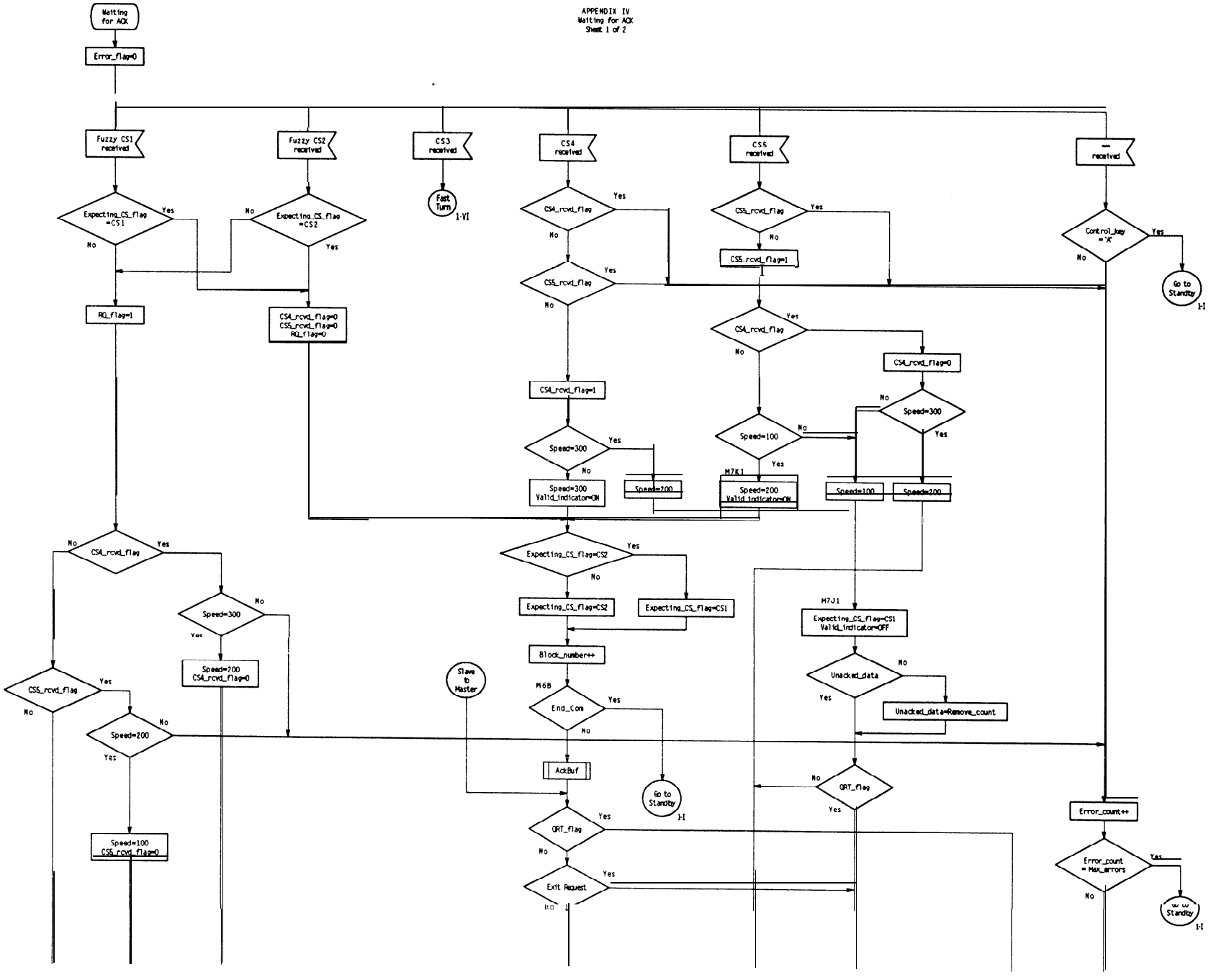


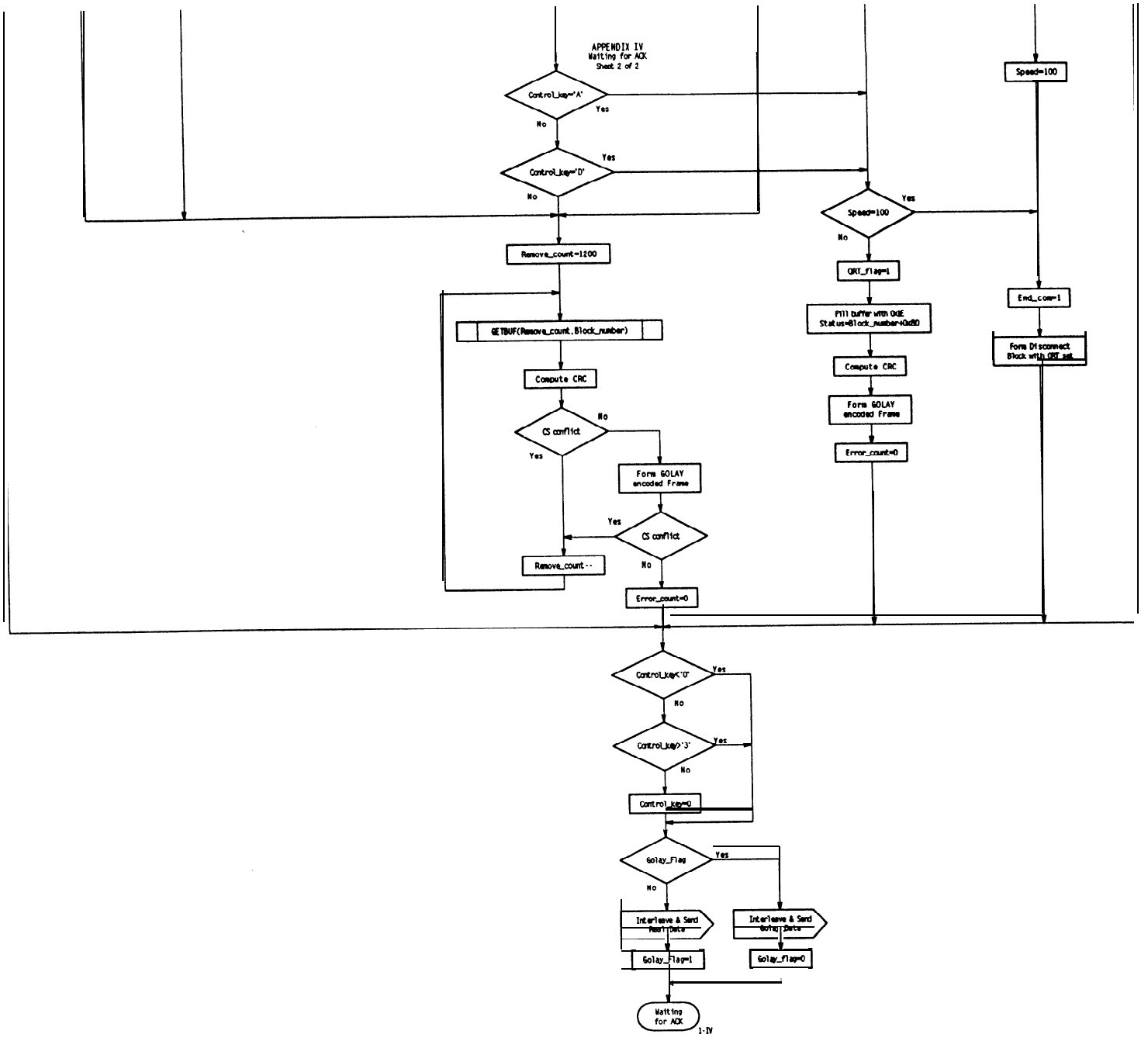
APPENDIX I I
Standby/Waiting for receive
Sheet 1 of 1

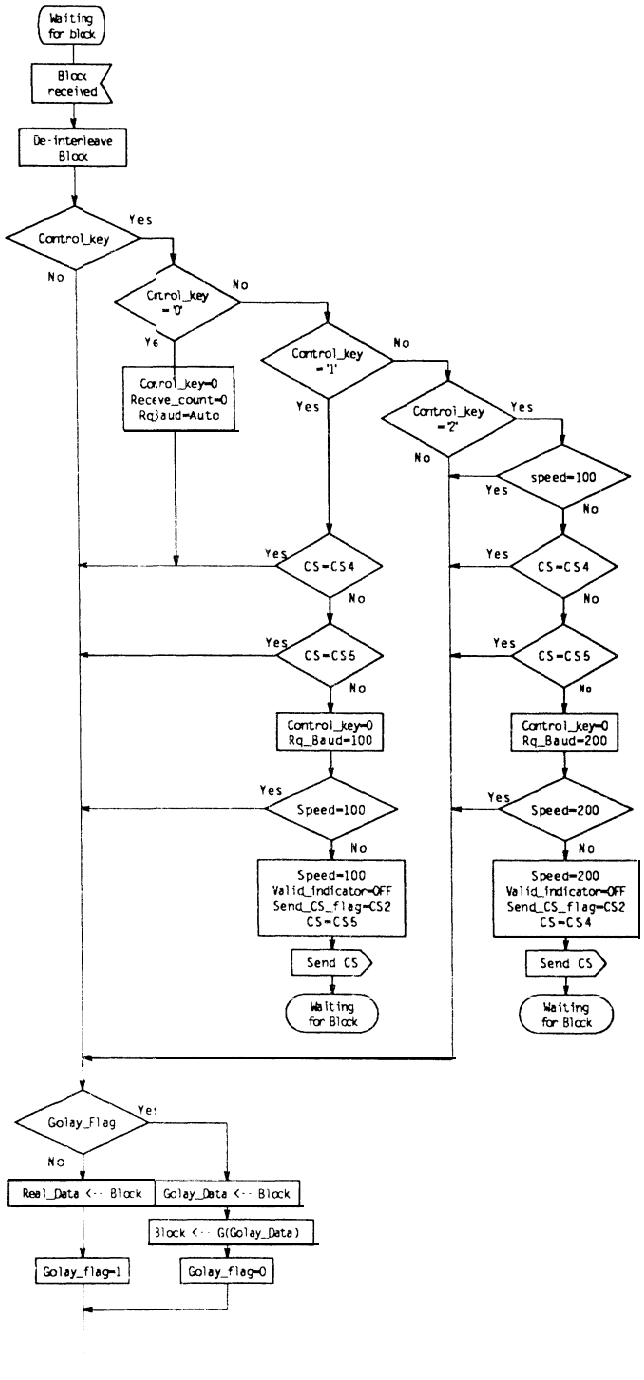


APPENDIX III
Connect in progress
Sheet 1 of 1

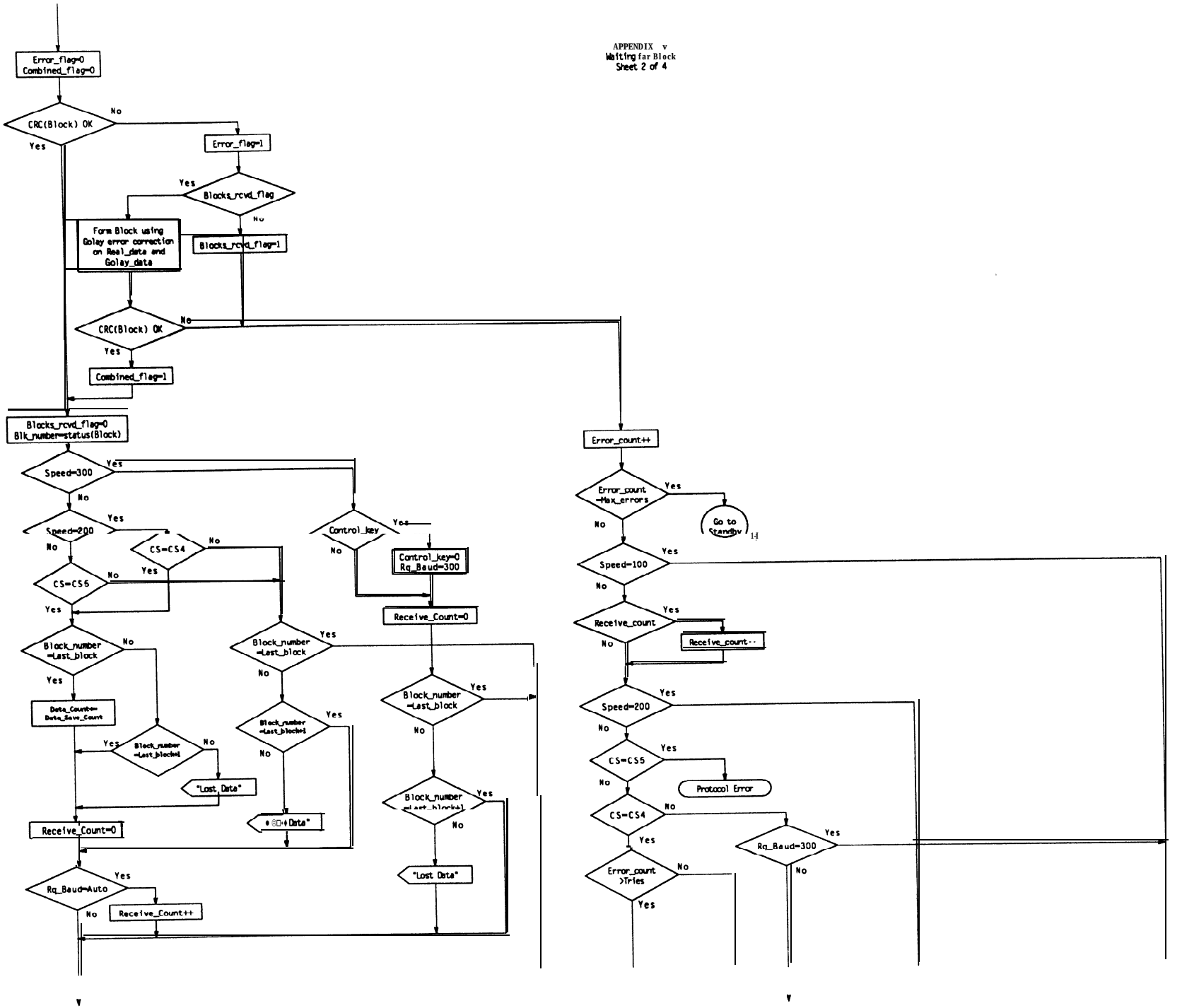


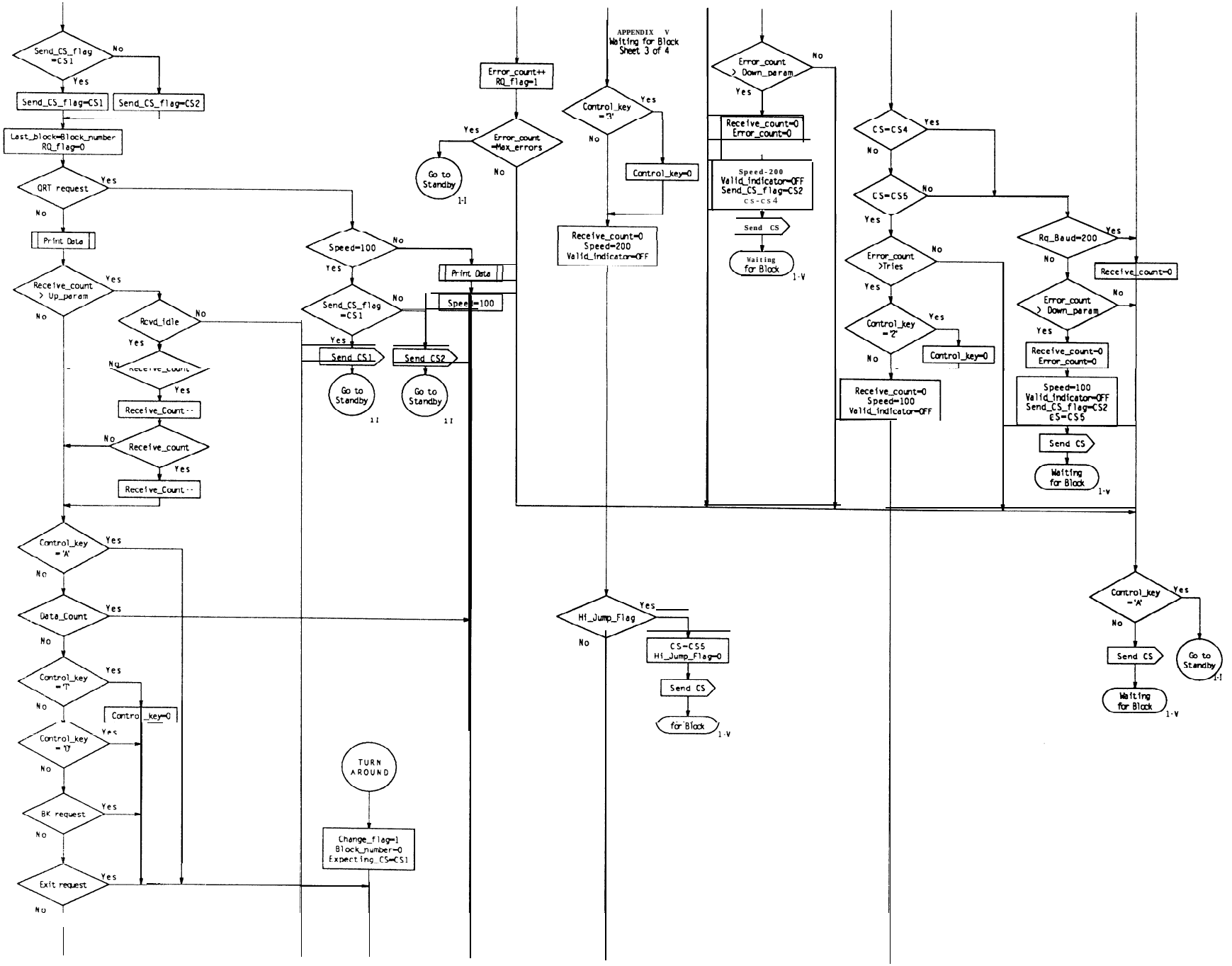

APPENDIX IV
Waiting for ACK
Sheet 1 of 2
Waiting for ACK
Error_flag=0
Fuzzy CS1 received
Fuzzy CS2 received
CS3 received
CS4 received
CS5 received
CS6 received
Fast Turn
1-VI
Expecting_CS_flag =CS1
Expecting_CS_flag =CS2
CS4_rcvd_flag
CS5_rcvd_flag
CS5_rcvd_flag=1
RQ_flag=1
CS4_rcvd_flag=0 CS5_rcvd_flag=0 RQ_flag=0
CS4_rcvd_flag=1
CS4_rcvd_flag=0
Speed=300
Speed=100
Speed=300 Valid_indicator=ON
Speed=200
M7K1
Speed=200 Valid_indicator=ON
Speed=100
Speed=200
Expecting_CS_flag=CS2
Expecting_CS_flag=CS2
Expecting_CS_flag=CS1
M7J1
Expecting_CS_flag=CS1 Valid_indicator=OFF
Block_number++
M6B
End_Com
Unacked_data
Unacked_data=Remove_count
Slave to Master
AckBuf
Go to Standby
I-I
ORT_flag
Exit Request
Error_count++
Error_count = Max_errors
Speed=100 CS5_rcvd_flag=0
Speed=200 CS4_rcvd_flag=0
Yes
No


APPENDIX IV
Waiting for ACK
Sheet 2 of 2
Control_key='A'
Yes
No
Control_key='D'
Yes
No
Remove_count=1200
GETBUF(Remove_count.Block_number)
Compute CRC
CS conflict
No
Yes
Form GOLAY encoded Frame
CS conflict
Yes
No
Remove_count--
Error_count=0
Speed=100
Yes
No
CRT_flag=1
Fill buffer with CKIE Status=Block_number+0x80
Compute CRC
Form GOLAY encoded Frame
Error_count=0
Speed=100
End_com=1
Form Disconnect Block with CRT set
Control_key<'0'
Yes
No
Control_key>'3'
Yes
No
Control_key=0
Golay_Flag
Yes
No
Interleave & Send Golay Data
Golay_flag=0
Interleave & Send Next Data
Golay_flag=1
Waiting for ACK
1-IV


Waiting for block
Block received
De-interleave Block
Control_key
Yes
No
Control_key = '0'
Yes
No
Control_key=0
Receive_count=0
Rq_baud=Auto
Control_key = '1'
Yes
No
Control_key = '2'
Yes
No
speed=100
Yes
No
CS=CS4
Yes
No
CS=CS5
Yes
No
Control_key=0
Rq_Baud=100
Speed=100
Yes
No
Speed=100
Valid_indicator=OFF
Send_CS_flag=CS2
CS=CS5
Send CS
Waiting for Block
CS=CS4
Yes
No
CS=CS5
Yes
No
Control_key=0
Rq_Baud=200
Speed=200
Yes
No
Speed=200
Valid_indicator=OFF
Send_CS_flag=CS2
CS=CS4
Send CS
Waiting for Block
Golay_Flag
Yes
No
Real_Data <-- Block
Golay_Data <-- Block
Block <-- G(Golay_Data)
Golay_flag=1
Golay_flag=0

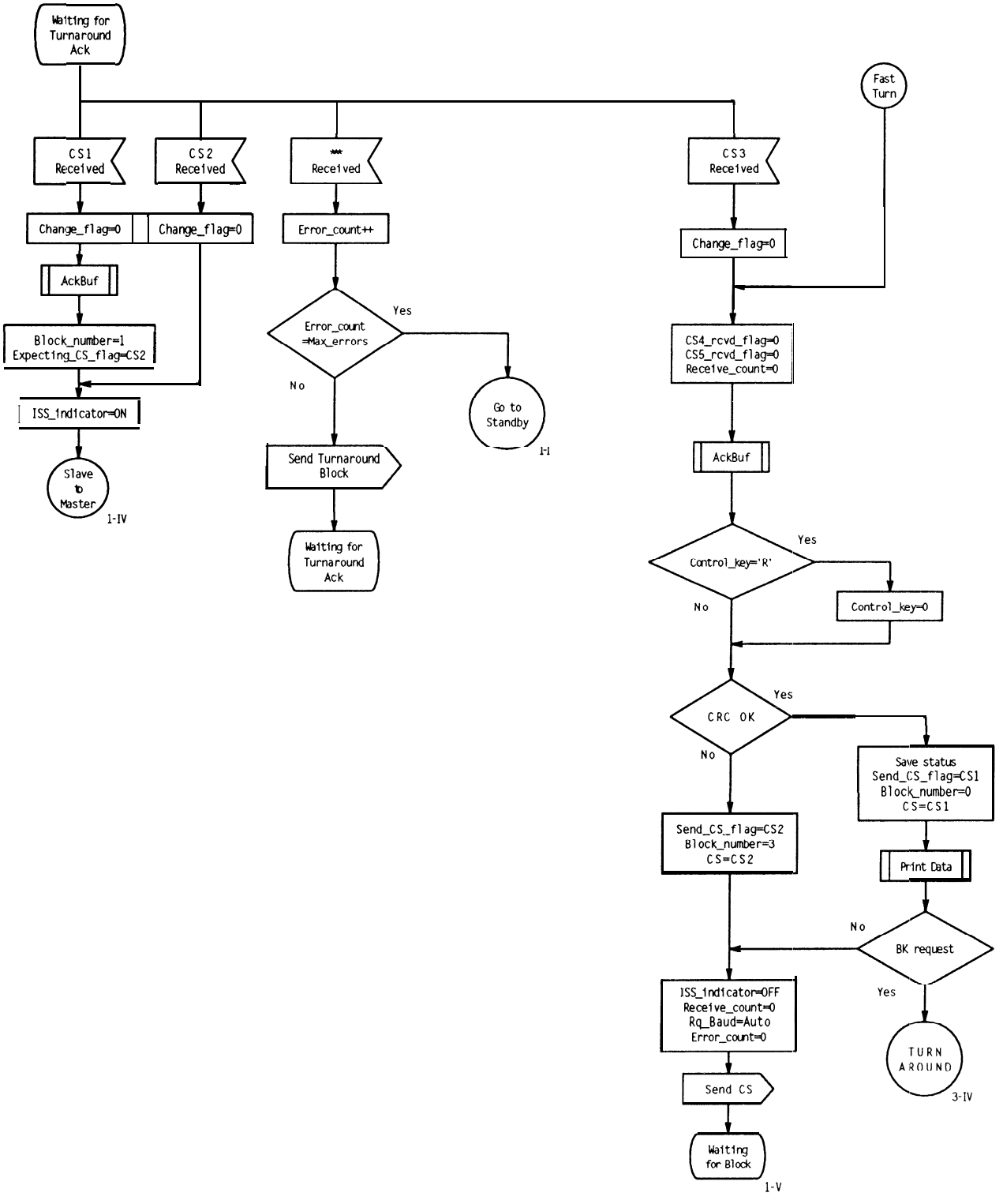
APPENDIX v
Waiting far Block
Sheet 2 of 4



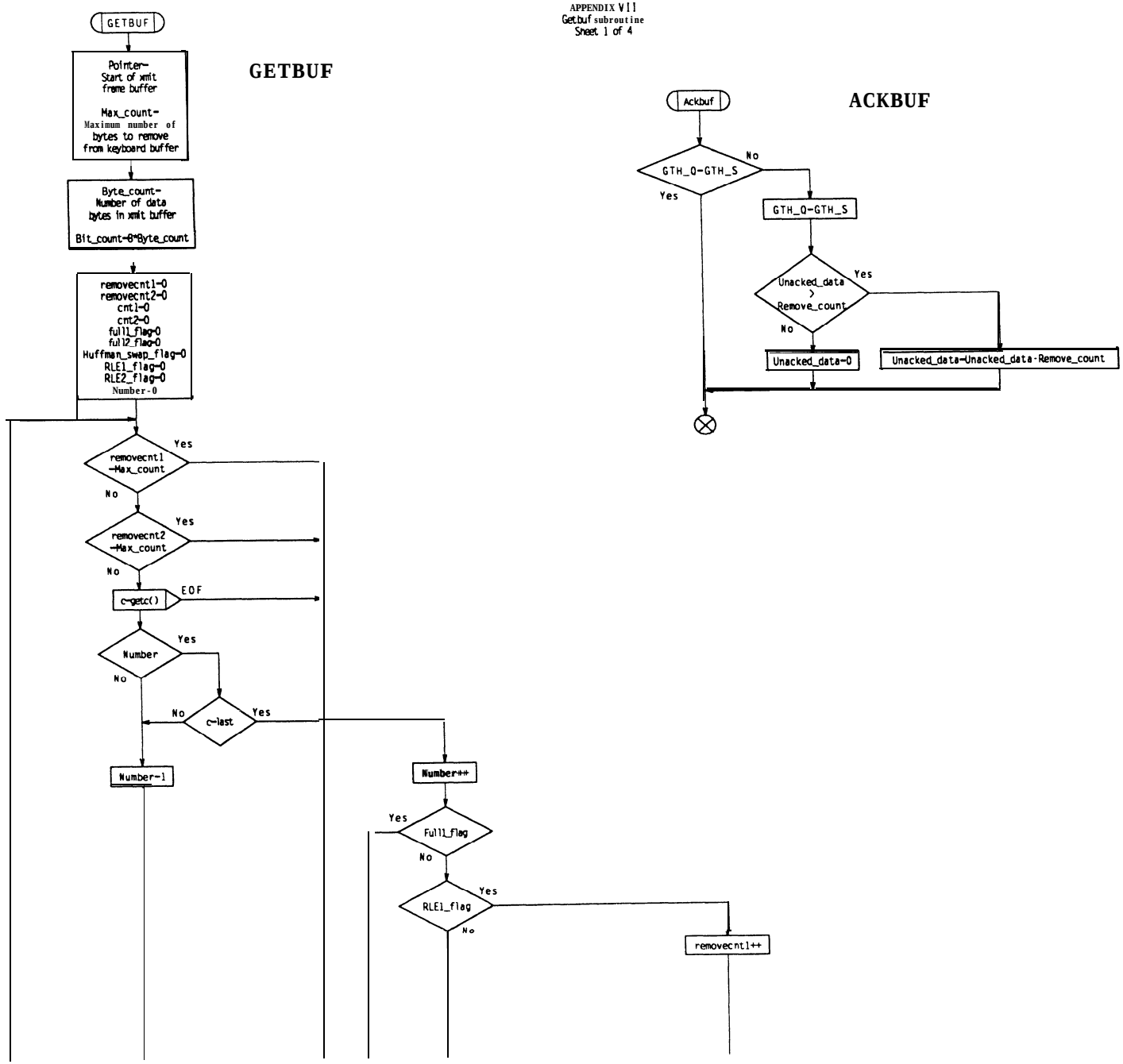

APPENDIX V
Waiting for Block
Sheet 3 of 4
Send_CS_flag =CS1
Send_CS_flag=CS1
Send_CS_flag=CS2
Last_block=Block_number
RQ_flag=0
QRT request
Print Data
Receive_count > Up_param
Rcvd_idle
Receive_Count--
Receive_count
Control_key = 'A'
Data_Count
Control_key = 'T'
Control_key=0
Control_key = 'D'
BK request
Exit request
Speed=100
Send_CS_flag =CS1
Send CS1
Send CS2
Go to Standby
Error_count++
RQ_flag=1
Error_count =Max_errors
Control_key = 'D'
Control_key=0
Receive_count=0
Speed=200
Valid_indicator=OFF
Hi_Jump_Flag
CS=CS5
Hi_Jump_Flag=0
Send CS
for Block
TURN AROUND
Change_flag=1
Block_number=0
Expecting_CS=CS1
Error_count > Down_param
Receive_count=0
Error_count=0
Speed-200
Valid_indicator=OFF
Send_CS_flag=CS2
CS-CS4
Waiting for Block
CS=CS4
CS=CS5
Error_count >Tries
Control_key = '2'
Receive_count=0
Speed=100
Valid_indicator=OFF
Rq_Baud=200
Receive_count=0
Error_count > Down_param
Receive_count=0
Error_count=0
Speed=100
Valid_indicator=OFF
Send_CS_flag=CS2
ES=CS5
Send CS
Waiting for Block
Control_key = 'A'
Send CS
Waiting for Block
Go to Standby
1-I
1-V

APPENDIX V
Waiting for Block
Sheet 4 of 4
Form Turnaround Block
Send Turnaround Block
Waiting for Turnaround Ack
1-VI
Control_key -2
No
Yes
CS-CS4
CS-CS5
Speed-100
Error_count-0
Receive_count-0
Valid_Indicator-OK
CS-CS5
Speed-200
Send CS
Waiting for Block
1-V
Control_key -3
CS-CS3
CS-CS
Speed-30
Speed-10
HI_Jump_Flag-1
Error_cont-0
Receive_cnt-0
Valid_Indicator-OK
CS-C1
Speed-00
Send S
Waiti for Blk
Send_CS_flag - CS2
CS-CS2
CS-CS1
Error_count-0
Send CS
Waiting for Block

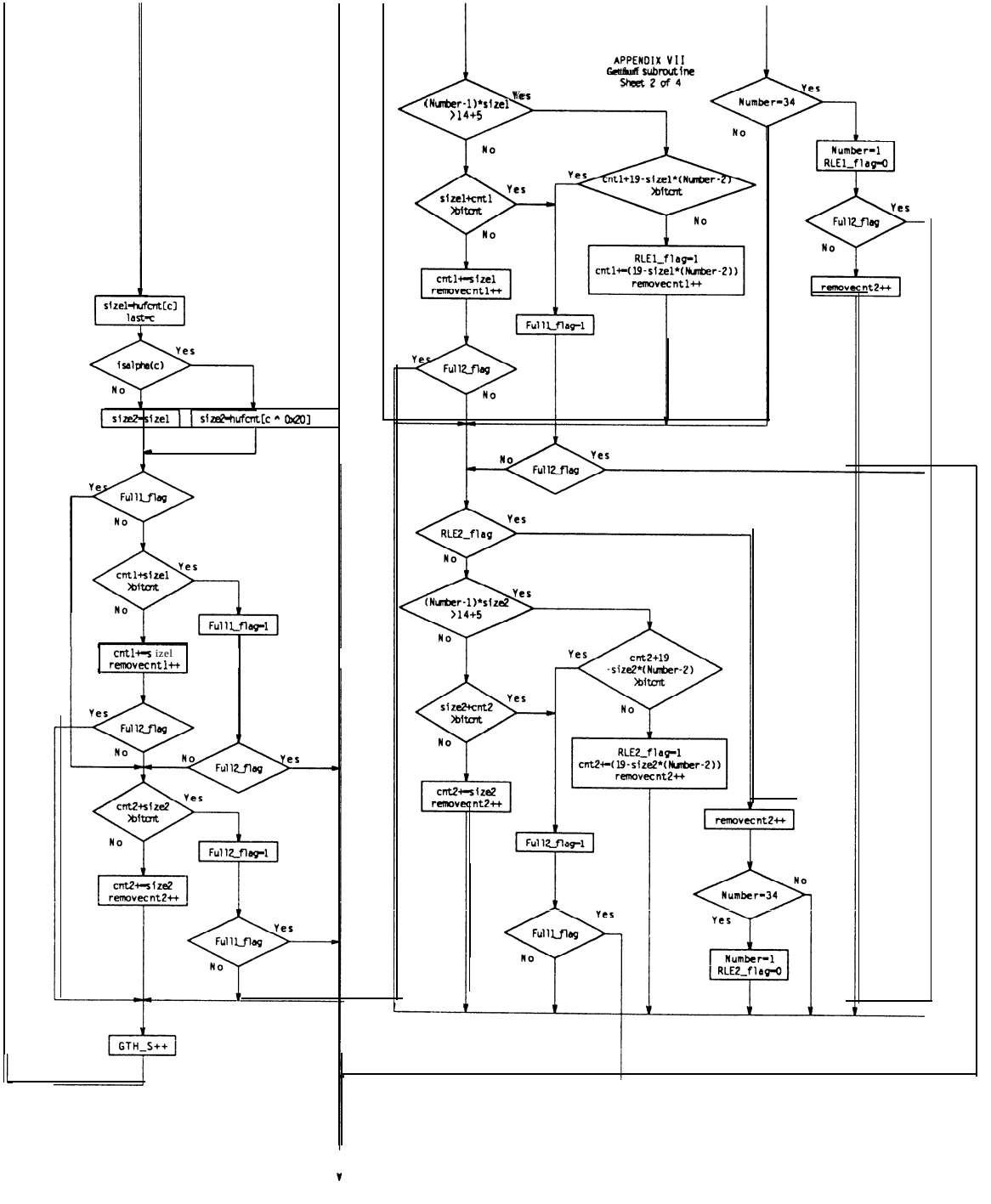
APPENDIX VI
Waiting for turnaround ACK
sheet 1 of 1



APPENDIX VII
Get.buf subroutine
Sheet 1 of 4

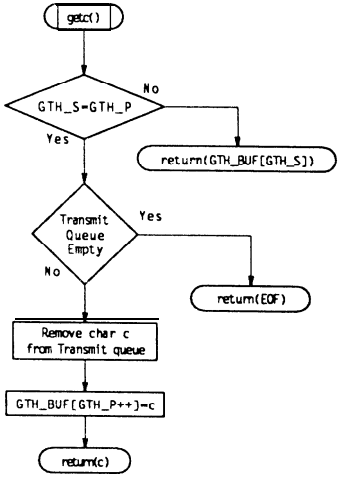


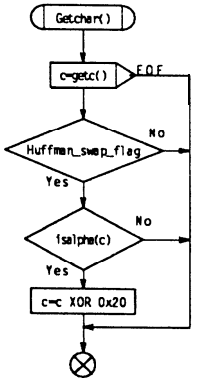
APPENDIX VII
Getbuff subroutine
Sheet 2 of 4

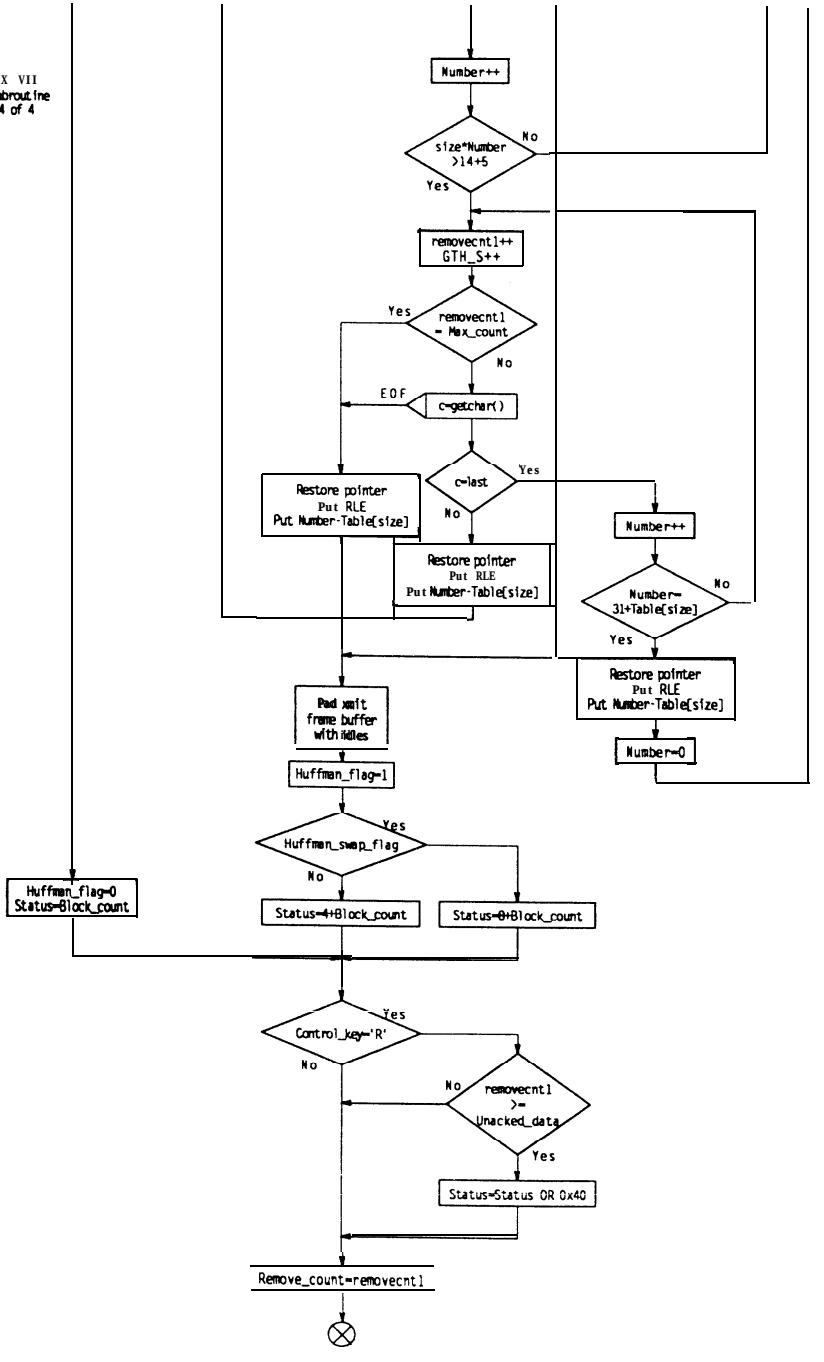


APPENDIX VII
Getbuf subroutine
Sheet 3 of 4

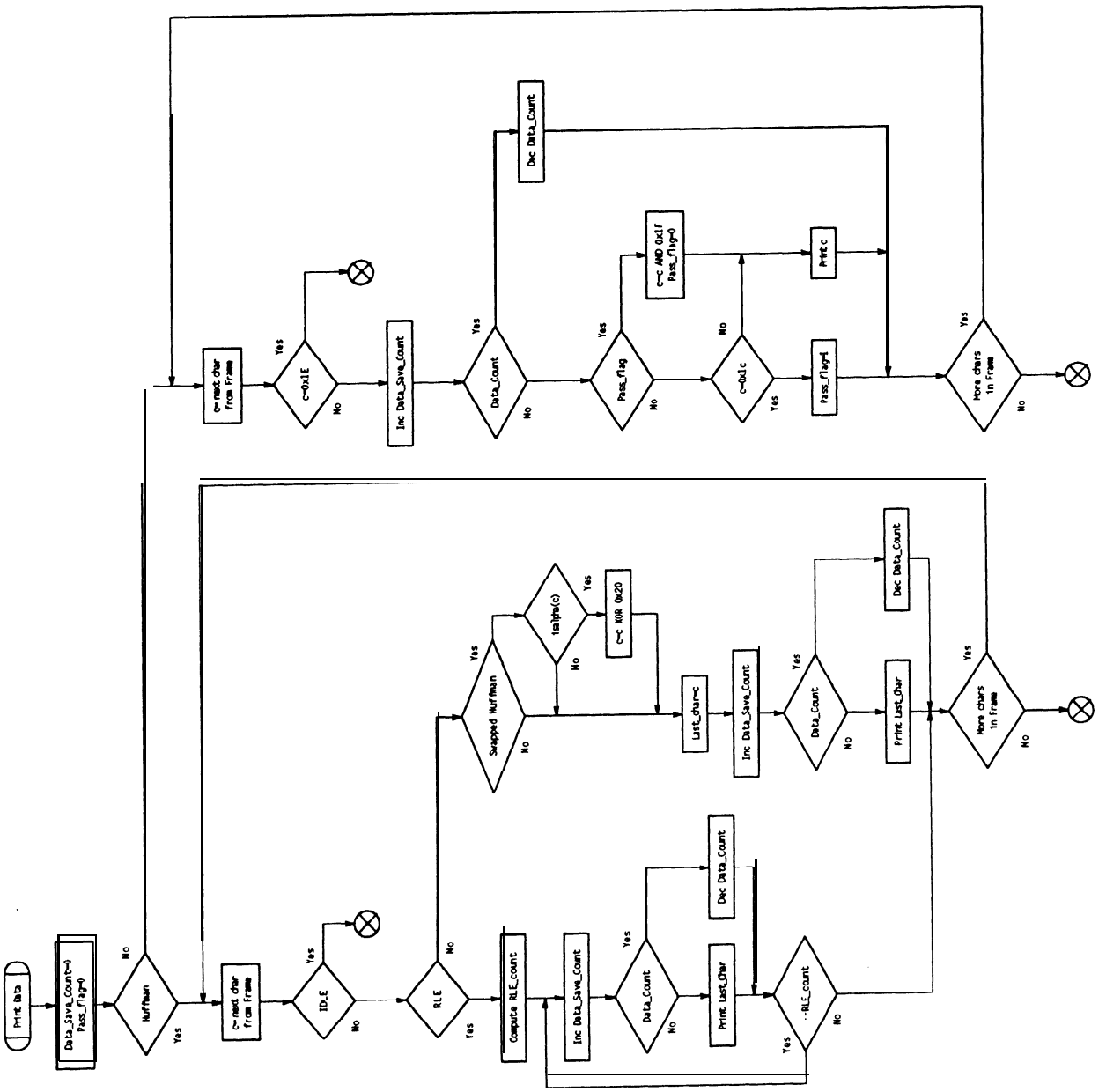
APPENDIX VII
Getbuf subroutine
Sheet 4 of 4

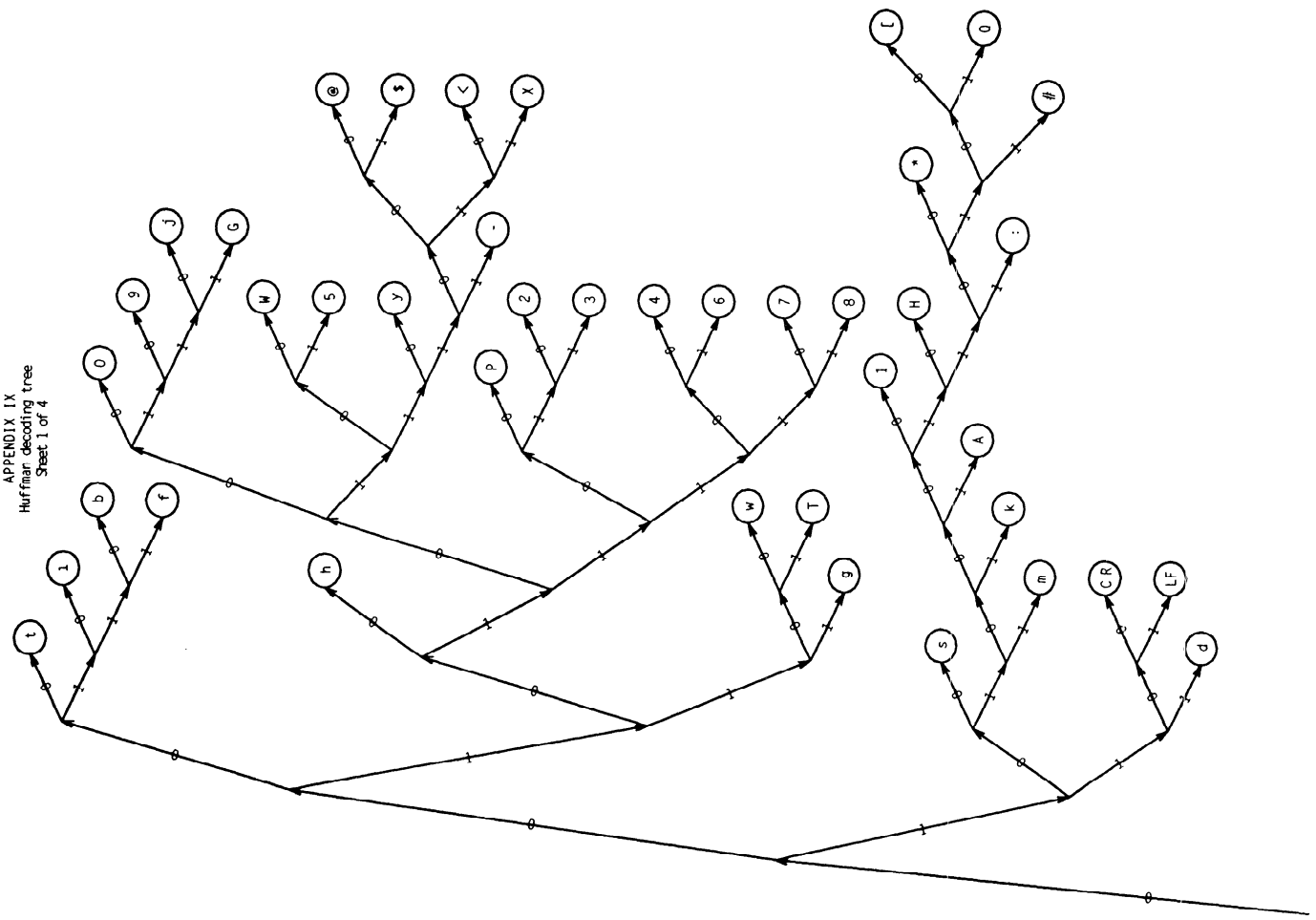




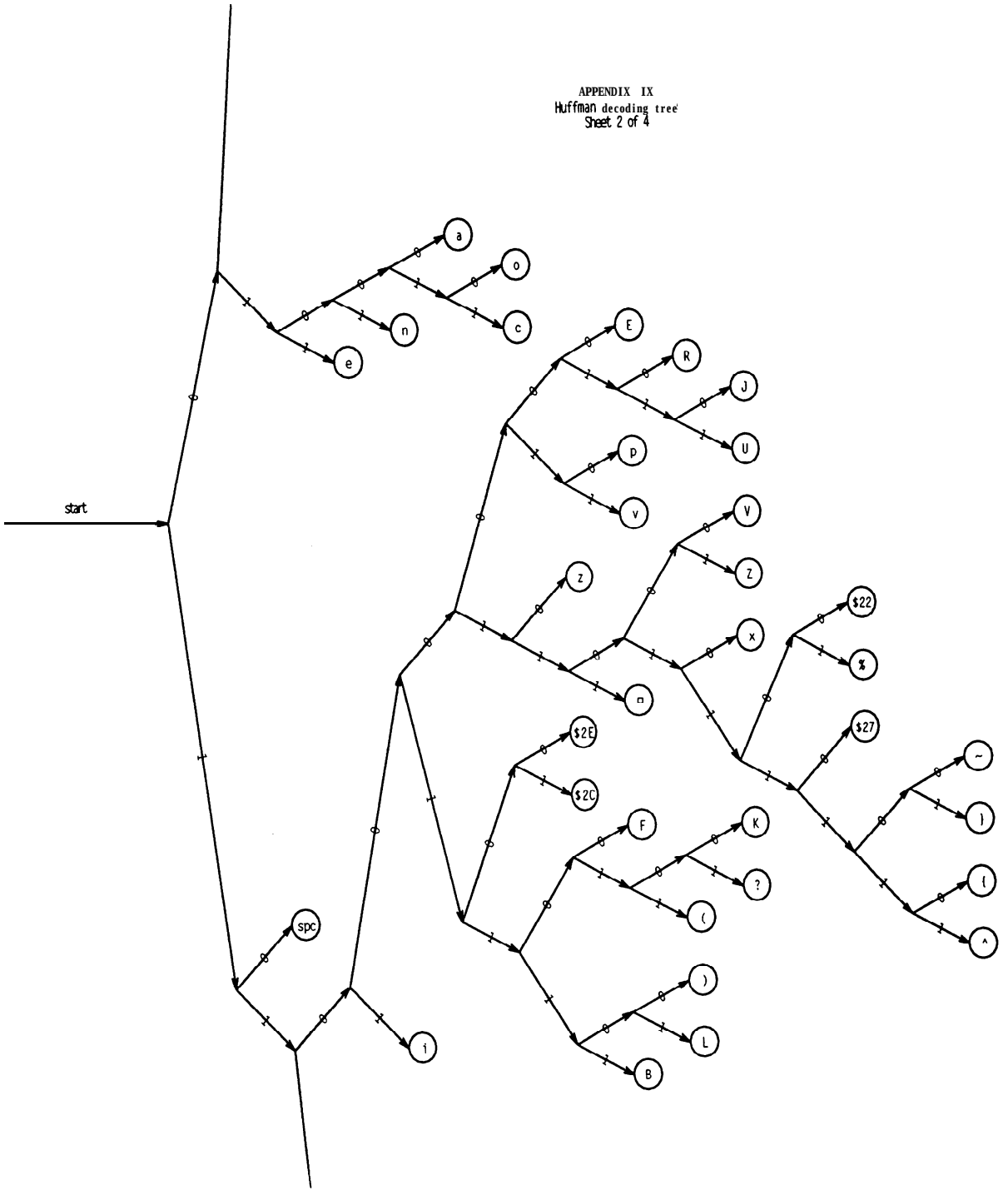


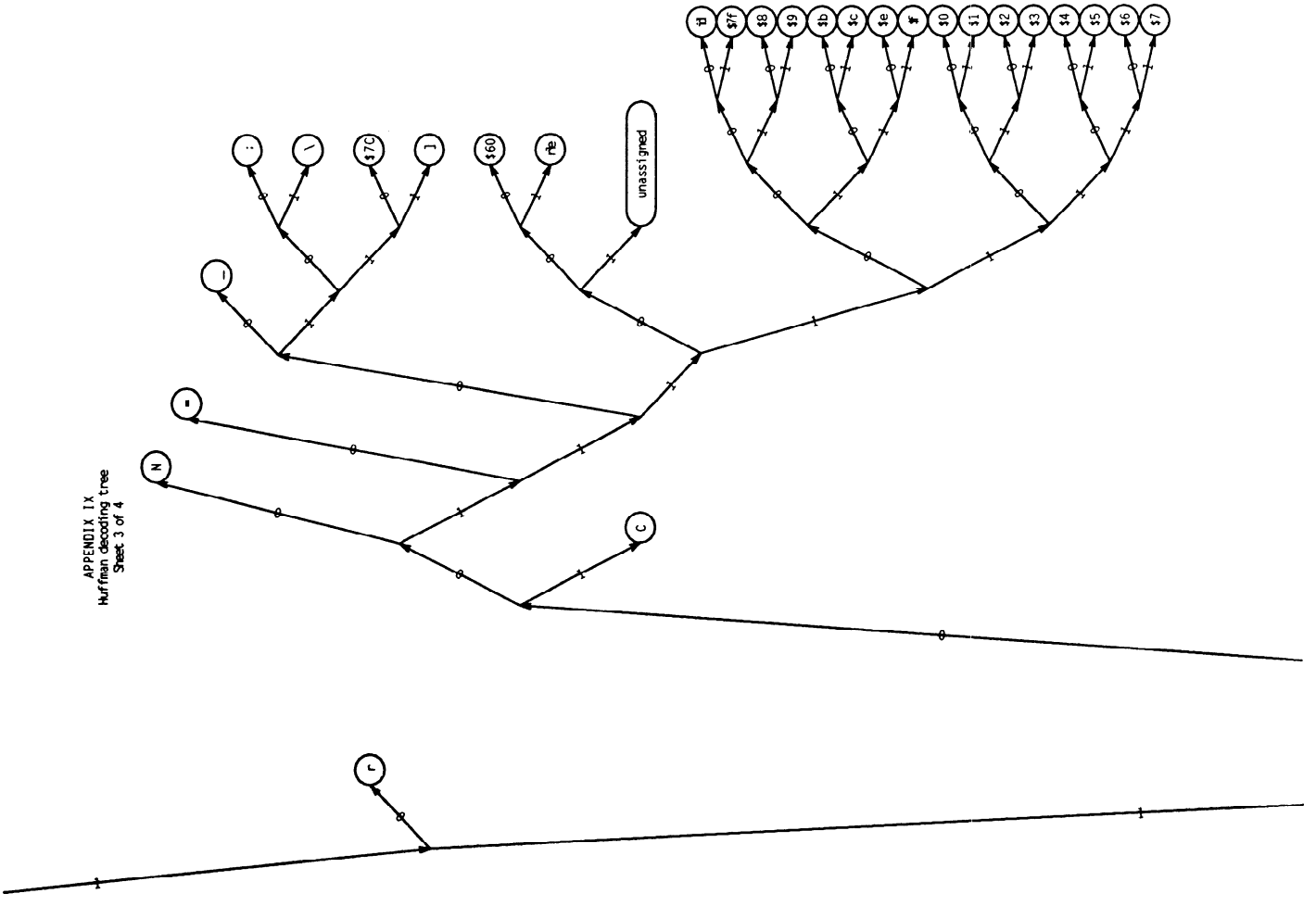
APPENDIX VIII
Print Data
Sheet 1 of 1

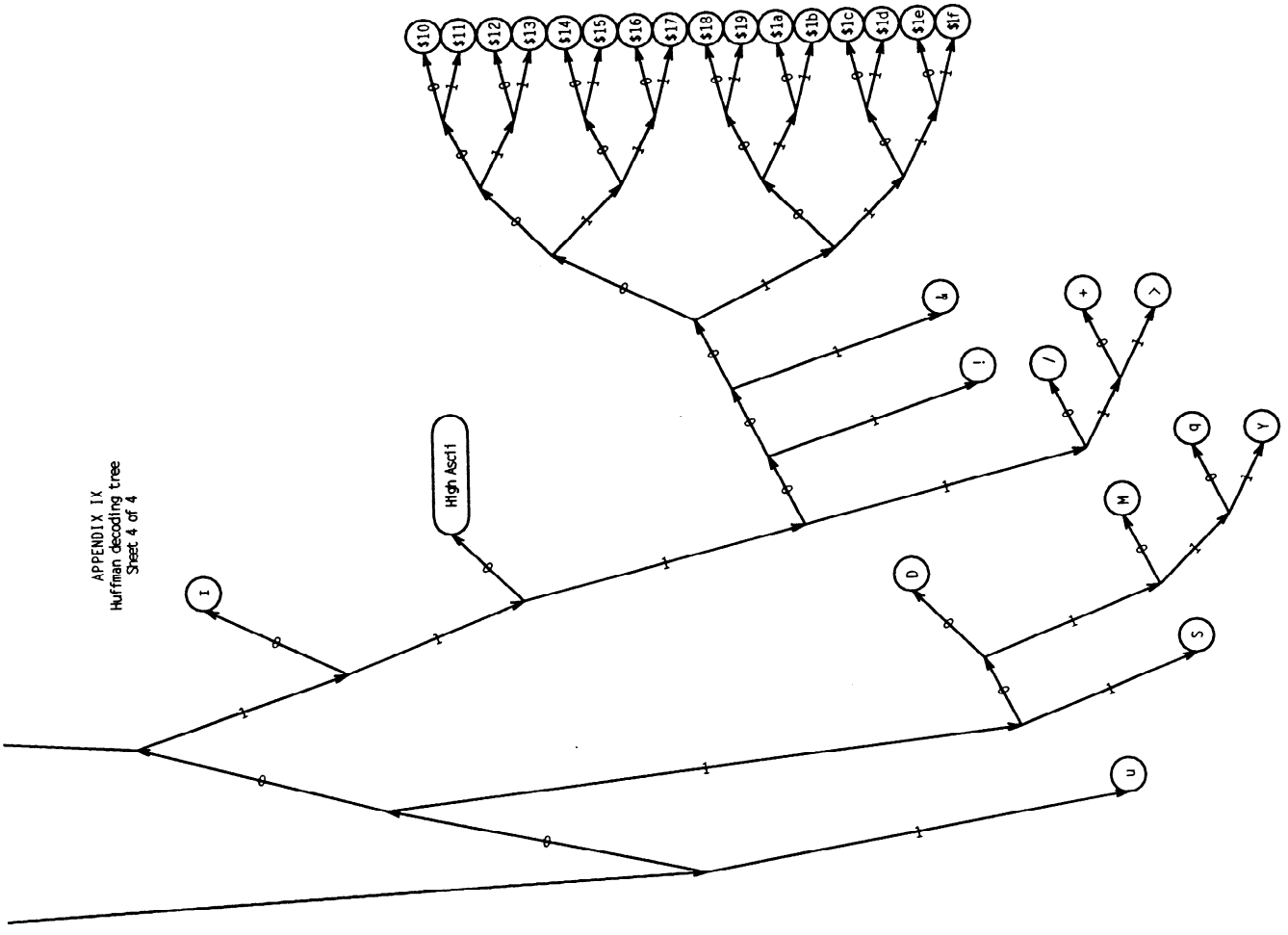



APPENDIX IX
Huffman decoding tree
Sheet 1 of 4

APPENDIX IX
Huffman decoding tree
Sheet 2 of 4




APPENDIX IX
Huffman decoding tree
Sheet 3 of 4
unassigned
$60
$7C
:
\
]
_
N
C
$f
$7f
$8
$9
$b
$c
$e
$0
$1
$2
$3
$4
$5
$6
$7


APPENDIX IX
Huffman decoding tree
Sheet 4 of 4
High Asc11
$10
$11
$12
$13
$14
$15
$16
$17
$18
$19
$1a
$1b
$1c
$1d
$1e
$1f
a
!
/
+
>
q
y
M
D
S
I
u

Appendix 10

C Program for **Golay** Encoding and Decoding

```
#include "stdlib. h"
#include "stdio. h"
#include "string. h"
#include "ctype. h"
unsigned g[4096],wt[4096];
unsigned b[12]=
   {0xDC5,0xB8B,0x717,0xE2D,0xC5B,0x8B7,
    0x16F,0x2DD,0x5B9,0xB71,0x6E3,0xFFE};
void create_golay_table(void)
{
unsigned i ,j ,data;
for(i=0;i<4096;i++)
   {
   for(j=0,data=0;j<12;j++)
      {
      if(i&(0x800>>j))data^=b[j];
      }
   g[i]=data;
   }
}
void   create-weight-table(void)
{
unsigned i,j ,data;
for(i=0;i<4096;i++)
   {
   for(j=0x800,data=0;j;j>>=1)
      {
      if(i&j)data++;
      }
   wt[i]=data;
   }
}
main(argc,argv)
int   argc;
char *argv[];
{
unsigned input,parity,i;
if(argc<2 I I argc>3 I I isalnum(argv[1][0])==0)
   {
   printf("g xxx       displays golay coding of "
                       "hex value xxx\n");
   printf("g xxx yyy   displays results of error "
                       "correction of xxx data "
                       "and yyy parity\n");
   return(O);
   }
if(sscanf(argv[1],"%x",&input)!=1)
   {
   printf("invalid data input\n");
   exit(l);
   }
if(input>0xFFF)
   {
   printf("input too large\n");
   exit(2);
   }
create_golay_table();
create-weight-tableo;
if(argc==2)printf("%3.3X ==> "
                  "%3.3X\n",input,g[input]);
else
   {
   if(sscanf(argv[2],"%x",&parity)!=1)
      {
      printf("invalid parity input\n");
      exit(3);
      }
   if(parity>0xFFF)
      {
      printf("parity too large\n");
      exit(4);
      }
   if(wt[input^g[parity]]<=3)
      {
      printf("%3.3X and %3.3X ==> "
             "%3.3X\n",input,parity,g[parity]);
      return(O);
      }
   for(i=0;i<12;i++)
      {
      if(wt[input^g[parity]^b[i]]<=2)
         {
         printf("%3.3X and %3.3X ==> "
                "%3.3X\n",
                input,parity,g[parity]^b[i]);
         return(O);
         }
      }
   if(wt[g[input]^parity]<=3)
      {
      printf("%3.3X and %3.3X ==> "
             "%3.3X\n",input,parity,input);
      return(0);
      }
 for(i=0;i<12;i++)
      {
      if(wt[g[input]^parity^b[i]]<=2)
         {
         printf("%3.3X and %3.3X ==> "
                "%3.3X\n",
                input,parity,input^(0x800>>i));
         return(O);
         }
      }
   printf("cannot correct\n");
   }
return(O);
}
```

Appendix 11

Huff man Table

by ASCII Code

0x00	1111000011111000	
0x01	1111000011111001	
0x02	1111000011111010	
0x03	1111000011111011	
0x04	1111000011111100	
0x05	1111000011111101	
0x06	1111000011111110	
0x07	1111000011111111	
0x08	1111000011110010	
0x09	1111000011110011	
0x0A	001101	
0x0B	1111000011110100	
0x0C	1111000011110101	
0x0D	001100	
0x0E	1111000011110110	
0x0F	1111000011110111	
0x10	1111001110000000	
0x11	1111001110000001	
0x12	1111001110000010	
0x13	1111001110000011	
0x14	1111001110000100	
0x15	1111001110000101	
0x16	1111001110000110	
0x17	1111001110000111	
0x18	1111001110001000	
0x19	1111001110001001	
0x1A	1111001110001010	
0x1B	1111001110001011	
0x1C	1111001110001100	
0x1D	1111001110001101	
0x1E	1111001110001110	
0x1F	1111001110001111	
0x20	10	;' '
0x21	11110011101	;!
0x22	110001101100	;"
0x23	0010100011011	;#
0x24	0001010111001	;$
0x25	110001101101	;%
0x26	111100111001	;&
0x27	110001101110	;'
0x28	110011011	;(
0x29	110011100	;)
0x2A	001010001100	;*
0x2B	111100111110	;+
0x2C	1100101	;,
0x2D	00010101111	;-
0x2E	1100100	;.
0x2F	11110011110	;/
0x30	11000111	;0
0x31	001010000	;1
0x32	0001011010	;2
0x33	0001011011	;3
0x34	0001011100	;4
0x35	0001010101	;5
0x36	0001011101	;6
0x37	0001011110	;7
0x38	0001011111	;8
0x39	0001010010	;9
0x3A	00101000111	;:
0x3B	11110000110100	;;
0x3C	0001010111010	;<
0x3D	1111000010	;=
0x3E	111100111111	;>
0x3F	1100110101	;?
0x40	0001010111000	;@
0x41	00101001	;A
0x42	11001111	;B
0x43	11110001	;C
0x44	11110100	;D
0x45	11000000	;E
0x46	11001100	;F
0x47	00010100111	;G
0x48	0010100010	;H
0x49	11110010	;I
0x4A	1100000110	;J
0x4B	1100110100	;K
0x4C	110011101	;L
0x4D	111101010	;M
0x4E	111100000	;N
0x4F	000101000	;O
0x50	000101100	;P
0x51	00101000110101	;Q
0x52	110000010	;R
0x53	1111011	;S
0x54	0001101	;T
0x55	1100000111	;U
0x56	1100011000	;V
0x57	0001010100	;W
0x58	0001010111011	;X
0x59	1111010111	;Y
0x5A	1100011001	;Z
0x5B	00101000110100	;[
0x5C	11110000110101	;\
0x5D	11110000110111	;]
0x5E	11000110111111	;^
0x5F	111100001100	;_
0x60	11110000111000	;`
0x61	01000	;a
0x62	0000110	;b
0x63	010011	;c
0x64	00111	;d
0x65	011	;e
0x66	0000111	;f
0x67	000111	;g
0x68	000100	;h
0x69	1101	;i
0x6A	00010100110	;j
0x6B	0010101	;k
0x6C	000010	;l
0x6D	001011	;m
0x6E	0101	;n
0x6F	010010	;o
0x70	11000010	;p
0x71	1111010110	;q
0x72	1110	;r
0x73	00100	;s
0x74	00000	;t
0x75	11111	;u
0x76	11000011	;v
0x77	0001100	;w
0x78	1100011010	;x
0x79	0001010110	;y
0x7A	1100010	;z
0x7B	11000110111110	;{
0x7C	11110000110110	;|
0x7D	11000110111101	;}
0x7E	11000110111100	;~
0x7F	1111000011110001	
111100110xxxxxxx		;upper ascii
0x80	1111001100000000	
0X81	1111001100000001	
0X82	1111001100000010	
etc.		
IDLE	1111000011110000	
RLE	11110000111001	
UNUSED	1111000011101	

Huffman Table
by Huffman Code

0x20	10	;' '
0x65	011	;e
0x69	1101	;i
0x6E	0101	;n
0x72	1110	;r
0x61	01000	;a
0x64	00111	;d
0x73	00100	;s
0x74	00000	;t
0x75	11111	;u
0x0A	001101	;LF
0x0D	001100	;CR
0x63	010011	;c
0x67	000111	;g
0x68	000100	;h
0x6C	000010	;l
0x6D	001011	;m
0x6F	010010	;o
0x2C	1100101	;,
0x2E	1100100	;.
0x53	1111011	;S
0x54	0001101	;T
0x62	0000110	;b
0x66	0000111	;f
0x6B	0010101	;k
0x77	0001100	;w
0x7A	1100010	;z
0x30	11000111	;0
0x41	00101001	;A
0x42	11001111	;B
0x43	11110001	;C
0x44	11110100	;D
0x45	11000000	;E
0x46	11001100	;F
0x49	11110010	;I
0x70	11000010	;P
0x76	11000011	;v
0x28	110011011	;(
0x29	110011100	;)
0x31	001010000	;1
0x4C	110011101	;L
0x4D	111101010	;M
0x4E	111100000	;N
0x4F	000101000	;O
0x50	000101100	;P
0x52	110000010	;R
0x32	0001011010	;2
0x33	0001011011	;3
0x34	0001011100	;4
0x35	0001010101	;5
0x36	0001011101	;6
0x37	0001011110	;7
0x38	0001011111	;8
0x39	0001010010	;9
0x3D	1111000010	;=
0x3F	1100110101	;?
0x48	0010100010	;H
0x4A	1100000110	;J
0x4B	1100110100	;K
0x55	1100000111	;U
0x56	1100011000	;V
0x57	0001010100	;W
0x59	1111010111	;Y
0x5A	1100011001	;Z
0x71	1111010110	;q
0x78	1100011010	;x
0x79	0001010110	;y
0x21	11110011101	;!
0x2D	00010101111	;-
0x2F	11110011110	;/
0x3A	00101000111	;:
0x47	00010100111	;G
0x6A	00010100110	;j
0x22	110001101100	;"
0x25	110001101101	;%
0x26	111100111001	;&
0x27	110001101110	;'
0x2A	001010001100	;*
0x2B	111100111110	;+
0x3E	111100111111	;>
0x5F	111100001100	;_
0x23	0010100011011	;#
0x24	0001010111001	;$
0x3C	0001010111010	;<
0x40	0001010111000	;@
0x58	0001010111011	;X
UNUSED	1111000011101	
0x3B	11110000110100	;;
0x51	001010001101 01	;Q
0x5B	001010001101 00	;[
0x5C	111100001101 01	;\
0x5D	111100001101 11	;]
0x5E	110001101111 11	;^
0x60	111100001110 00	;`
0x7B	110001101111 10	;{
0x7C	111100001101 10	;|
0x7D	110001101111 01	;}
0x7E	110001101111 00	;~
RLE	111100001110 01	
0x00	1111000011111000	
0x01	1111000011111001	
0x02	1111000011111010	
0x03	1111000011111011	
0x04	1111000011111100	
0x05	1111000011111101	
0x06	1111000011111110	
0x07	1111000011111111	
0x08	1111000011110010	
0x09	1111000011110011	
0x0B	1111000011110100	
oxoc	1111000011110101	
0x0E	1111000011110110	
0x0F	1111000011110111	
0x10	1111001110000000	
0x11	1111001110000001	
0x12	1111001110000010	
0x13	1111001110000011	
0x14	1111001110000100	
0x15	1111001110000101	
0x16	1111001110000110	
0x17	1111001110000111	
0x18	1111001110001000	
0x19	1111001110001001	
0x1A	1111001110001010	
0x1B	1111001110001011	
0x1C	1111001110001100	
0x1D	1111001110001101	
0x1E	1111001110001110	
0x1F	1111001110001111	
0x7F	1111000011110001	
111100110xxxxxxx		upperascii
0x80	1111001100000000	
0X81	1111001100000001	
OX82	1111001100000010	
etc.		
IDLE	1111000011110000	

For the Huffman decoding tree, see Appendix 9.

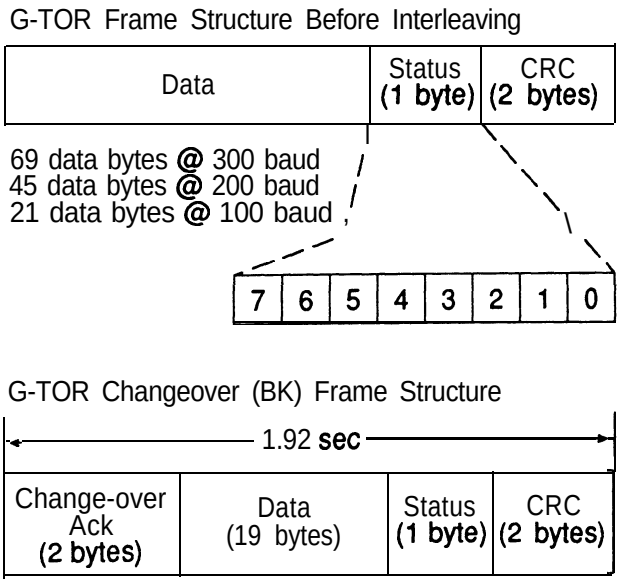
GMON – a G-TOR™ Monitoring Program for PC Compatibles

by Richard Huslig and Phil Anderson, WØXI

The G-TOR data communications protocol is an innovation of the technical staff of Kantronics Co., Inc. It was introduced in March 1994 as an inexpensive means of improving point-to-point digital communications in the HF radio bands. It has been implemented in the KAM Plus and KAM-E and is now offered for licensing to other manufacturers.

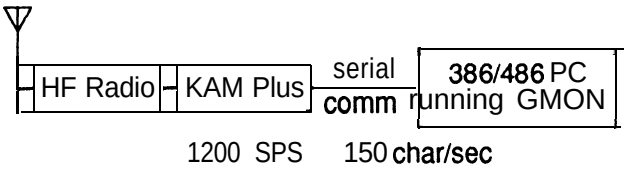
Monitoring of G-TOR frames is difficult since a variety of frame formats exist as shown in Figure 1. Although a data frame is always 1.92 seconds in duration, it might be sent at 100,200, or 300 baud; it might contain real data or Golay parity bits in ASCII or Huffman encoded form; and it may be received in Lower or Upper Side Band. Each data frame must be deinterleaved at the receiver and this consumes a large amount of processing time. Data may exist in BK (BreaK or changeover) frames too; although, they are sent at only 100 baud without interleaving. In addition, no synchronizing flags exist at the beginning or end of the frames. Each data frame contains data, a status byte, and two CRC bytes. The connect and disconnect frames, sent at only 100 baud, contain destination and source address fields in place of the data frame's data field.

Figure 1
G-TOR Frame Structures



Hence, monitoring on the fly – in real time – is best accomplished using a brute force algorithm with the system shown in Figure 2.

Figure 2
G-TOR Monitoring Setup



Frames are received by an HF radio, demodulated by the KAM Plus (or TNC), and processed in 1.92 second segments by GMON, a PC-based basic terminal program that runs on a 386 or 486 compatible and includes G-TOR monitoring capability. The KAM Plus is configured via the GSCAN command to sample the receiver modem digital data 1200 times per second. These data are then shipped continuously via the serial port to the PC where GMON carries out the monitor processing, looking for valid frames at each sample. This paper will describe the development of GMON, provide a functional description of the protocol, and detail the theory of GMON operation.

Background

To realize our goal of developing a terminal program to monitor all G-TOR frames, we first developed GOFF, a program that would monitor G-TOR frames from a file of sampled data. This program, written in BORLAND C, operated offline from the TNC. GOFF performed 100,200, and 300 baud tests and a 100 baud BK test on 1.92 seconds of modem data samples. The baud tests consisted of deinterleaving, packing, CRC checking, Golay decoding, and again CRC checking. The 100 baud BK test consisted of shifting and CRC checking. If the CRC was valid, then the status byte was decoded; if the frame was a connect, disconnect, data, or BK frame, the frame was suitable for display. If the frame was a data frame and the status byte also indicated Huffman coding, then Huffman and RLEn decoding was performed. All other frames with invalid CRC or status byte were discarded.

GOFF began as a brute force algorithm which performed all tests on every sample at 600 SPS (Samples Per Second). A 7027 byte (7027*8/600 = 93.7 sec.) data file took 140 sec. to process and display on a 50 MHz 486DX machine. After analyzing and optimizing the loops with the highest iterations, we cut this time to 105 sec. Then, after optimizing the assembly code generated from the C code, we cut it to 80 sec. Next, we realized that if we cut the number of tests by processing only 2 samples of each bit at each baud rate, we could cut the number of tests per sample from 4 to a little more than 1, even though we had to increase the sample rate to 1200 SPS. By phasing the tests, we cut the time to 60 sec. Next we optimized the deinterleave process by storing the state of the interleave buffer at each baud rate and each bit phase, so that to deinterleave, we simply rotate the interleave buffer and shift in the new sample. This cut the time to 29 sec. Now, removing the CRC test of the inverted data and optimizing the assembly code by using string instructions, we cut the time to 16 sec. Further optimization yielded 14 sec. Processing the same file on a 386DX-33 took 39 sec., and after turning the turbo switch off (i.e. 386DX-8), processing took 85 sec. Processing on a 286-16 took 111 sec. Further speed enhancement is possible if the trace option is turned off. Or, if a valid G-TOR frame is detected by a valid CRC, most processing, except deinterleaving and shifting, on the next 1.92 sec. of data, can be skipped. Also if the same G-TOR frame is detected again by means of the second bit phase or a retransmitted frame which was not acknowledged, only the first frame is displayed.

After the GOFF program was debugged and optimized, then we developed the GMON program by integrating GOFF with a terminal program written in C. Later, we reworked GMON's modules, especially the G-TOR monitoring modules, to follow a specified calling convention so that the terminal program modules handled all data display and logging. The main G-TOR monitoring module was written in BORLAND C, and, therefore, follows the BORLAND C calling convention. The other 2 modules were written in assembly. These 3 object modules were combined into a library file called GMONTER.LIB, which can be linked with any other terminal program which adheres to the BORLAND C calling convention.

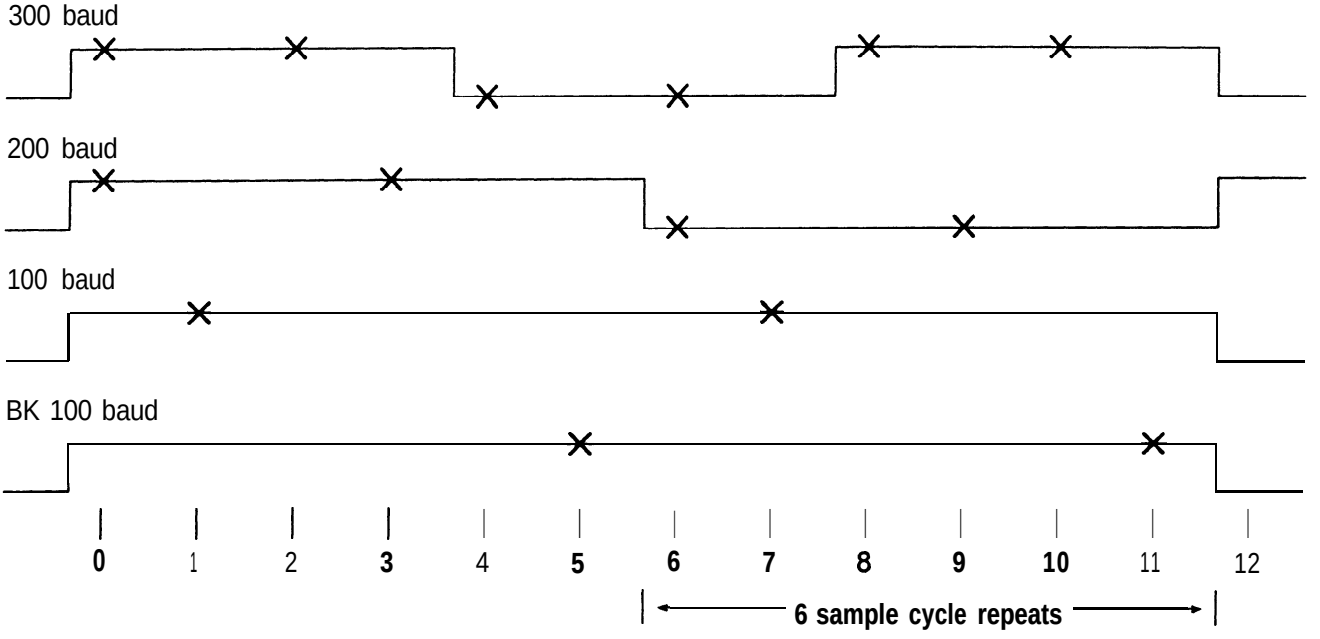
Functional **Description** of **GMON**

Sampling is initiated by issuing the GSCAN 1200 command to the KAM. When 8 samples are collected, the character is sent over the RS232 communication port at a constant 150 characters per second. Again, Figure 2 illustrates the G-TOR Monitoring Setup.

GMON receives the character in its interrupt driven receive buffer. If the receive buffer is not empty, the 8 samples are demultiplexed and stored in the sample buffer – one sample per byte.

The sampling and phasing process, illustrated in Figure 3, demonstrates that only 2 samples of the bit are tested at each baud rate and that, typically, only 1 test is performed per sample. The samples are either shifted and stored in one of two BK frame buffers or deinterleaved at 100, 200, or 300 baud and stored in one of 6 interleave buffers depending on the baud rate and bit sample phase. GMON does a 100 baud test every 6th sample (sample phase 1), a 200 baud test every 3rd sample (sample phases 0 and 3), a 300 baud test every other sample (sample phases 0, 2, and 4), and a 100 baud BK test every 6th sample (sample phase 5).

Figure 3
GMON Sampling and Phasing (x marks test performed at sample specified)



This results in a 6 sample cycle. After 1.92 seconds of data have been collected, the samples are either shifted and stored (if sample phase = 5) or deinterleaved and stored in their respective buffers. The order and steps of processing these frames are displayed in Figure 4, the GMON flow chart.

If the sample phase is not 5, then deinterleaved data is packed into a frame buffer. The CRC is calculated from the frame and compared with the CRC stored in the frame. If the CRCs match, then the frame buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the frame buffer is ready to be sent to the video monitor. If the CRCs do not match, then the frame buffer is Golay decoded, stored in the Golay buffer, and CRC tested. If the CRCs match, then the Golay buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the Golay buffer is ready to be sent to the video monitor. If the CRCs do not match, GMON discards the frame. GMON does not attempt to do Golay error correction.

If the sample phase is 5, then the CRC of one of the BK frame buffers (selected by the bit phase) is calculated from the frame and compared with the CRC stored in the frame. If the CRCs match, then the frame buffer is Huffman decoded if the status byte indicates Huffman encoding. Now the frame buffer is ready to be sent to the video monitor. If the CRCs do not match, GMON discards the frame.

Theory of Operation

In this section, the ideas outlined above are expanded in detail.

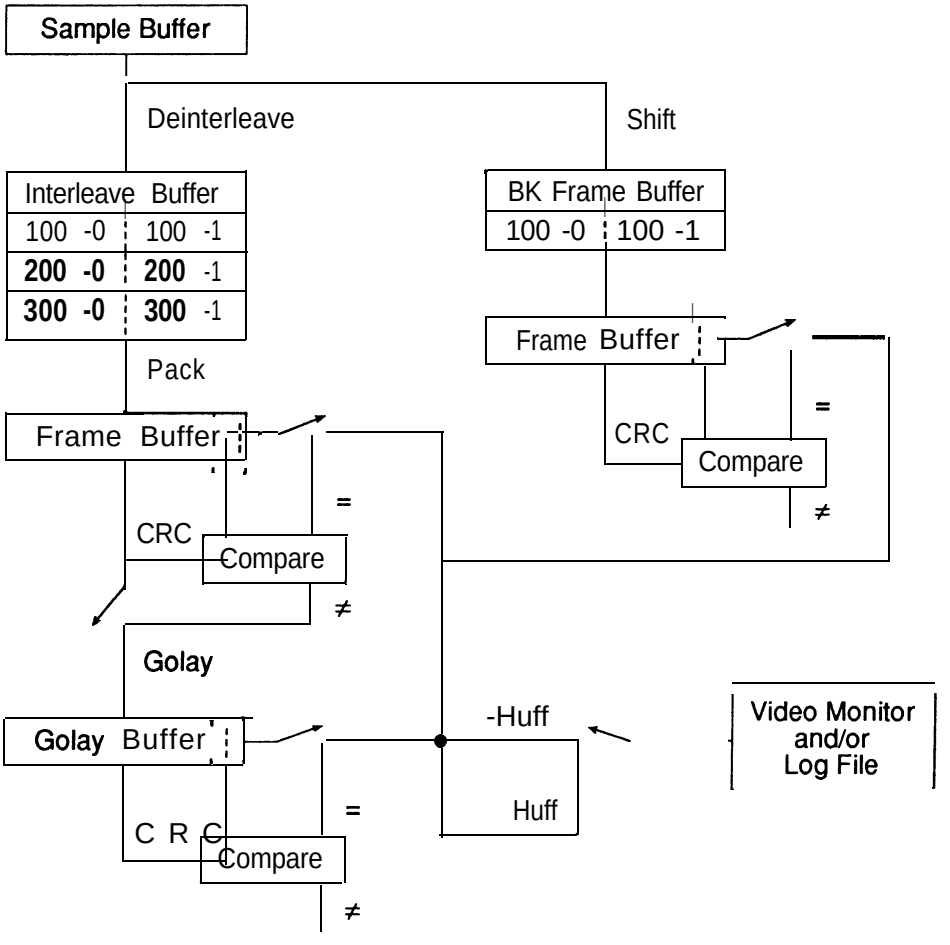
Sampling

GMON receives a character of 8 samples of the modem's data in its interrupt driven receive buffer at 150 characters per second. The 8 samples are demultiplexed and stored in the sample buffer – one sample per byte.

Phasing

Phasing reduces the number of tests per sample and, therefore, increases GMON's speed. GMON processes only 2 samples of the bit at each baud rate. Therefore of the 12 samples every 10 ms, GMON does a 100 baud test on

Figure 4
GMON Flow Chart



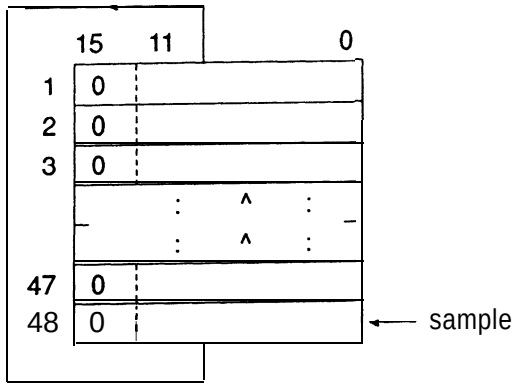
2 of the 12 samples, a 200 baud test on 4 of the 12 samples, a 300 baud test on 6 of the 12 samples, and a 100 baud BK test on 2 of the 12 samples. This establishes a 6 sample cycle:

Sample	Baud Test
6*n+0	300 and 200
6*n+1	100
6*n+2	300
6*n+3	200
6*n+4	300
6*n+5	100 BK

Deinterleaving

GMON's deinterleaving scheme significantly reduces the time needed to deinterleave 1.92 seconds of data. Instead of shifting all samples (576,384,192 for 300,200,100 baud respectively) for each new sample, this scheme, illustrated in Figure 5, rotates the interleave buffer and shifts in only the new sample; each 12-bit word shift is equivalent to 12 bit shifts. The samples are deinterleaved at 100, 200, or 300 baud and stored in one of 6 interleave buffers depending on the baud rate and bit sample phase. Each 300 baud interleave buffer contains 48 12-bit words, while each 200 baud interleave buffer contains 32 12-bit words, and each 100 baud interleave buffer contains 16

Figure 5
Interleave Buffer

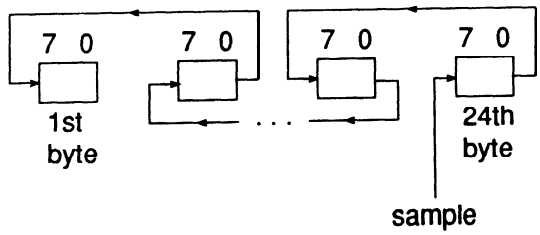


12-bit words. The interleave buffering scheme arranges the samples, which are exactly one bit time away, to be adjacent to one another in the same buffer. This scheme simplifies the deinterleaving task by simply shifting the words up one, bringing the top word down to the bottom and left shifting only the new sample into the LSB of the last word.

Shifting

The bit shifting task for BK frames, illustrated in Figure 6, simply shifts samples into one of two frame buffers depending on the bit sample phase. Since the LSB of the BK frame's first byte is transmitted first, the new sample is right shifted into the MSB of the 24th byte. After 192 bits are shifted in, the first bit occurs at the LSB of the first word.

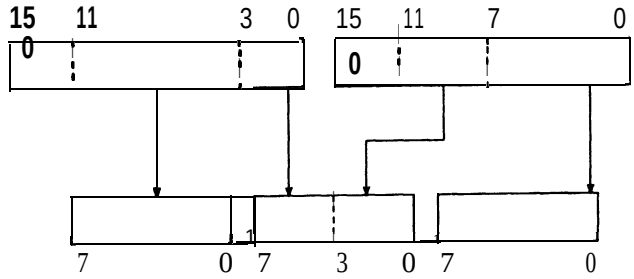
Figure 6
BK Frame Buffer



Packing

The packing task, illustrated in Figure 7, merely packs or formats each pair of 12-bit interleaved data words into three 8 bit bytes in the frame buffer. The 48 12-bit words of the 300 baud interleave buffers pack into 72 bytes of the frame buffer. The 32 12-bit words of the 200 baud interleave buffers pack into 48 bytes of the frame buffer. The 16 12-bit words of the 100 baud interleave buffers pack into 24 bytes of the frame buffer.

Figure 7
Packing 2 12-bit interleave words to 3 frame bytes



CRC Checking

Next the CRC is calculated from the frame and compared with the CRC stored in the frame. Much of the CRC calculation has been reduced to simple table lookup. If the CRCs match, then the frame buffer is tested for Huffman encoding. If the CRCs do not match, then the frame buffer is Golay decoded, stored in the Golay buffer, and CRC tested. If the CRCs match, then the Golay buffer is tested for Huffman encoding. If the CRCs do not match, GMON discards the frame.

Huffman Decoding

The frame or Golay buffer is Huffman decoded if the status byte indicates Huffman encoding. Huffman compressed data frames may contain RLEn (Run Length Encoding) codes. An RLEn code is a 19 bit code made up of a unique 14 bit Huffman code followed by 5 bits which represent a number n, O-31. When an RLEn code is encountered in a data frame, the previous character decoded in the frame should be repeated an additional N times where N is a number which depends on n and the number of bits used by the previous Huffman character. Refer to the G-TOR PROTOCOL for the Huffman table as well as the RLEn coding table.

Golay Decoding

If the CRCs do not match, then the frame buffer is Golay decoded and stored in the Golay buffer. The Golay decoding has been reduced to table lookup for quicker processing. The 4K 12-bit words of this table were generated by matrix multiplication of all combinations of 12-bit word inputs by the Golay Generator Matrix:

FFE
6E3
B71
5B9
2DD
16F
8B7
C5B
E2D
717
B8B
DC5

Summary

In conclusion the GMON terminal program efficiently monitors G-TOR frames using the fastest assembly language techniques. It also reduces the number of tests by testing only 2 samples of each bit. The deinterleave process is simplified by storing the state of the interleave buffer at each of 3 baud rates and each of 2 bit phases, and therefore only the current sample is shifted in after the interleave buffer is rotated. The CRC and Golay decoding is simplified by table lookup.

GMON's companion offline G-TOR monitoring program, GOFF, can also monitor G-TOR frames from sampled data stored in a disk file. This program is needed for slower PC compatible machines like the PC-XT. Kantronics Co., Inc. also provides GMONTER.LIB, a library of BORLAND C++ compatible object modules, which an external program can call to do the G-TOR monitoring process.

A Theoretical Evaluation of the G-TOR Hybrid ARQ Protocol

Glenn E. Prescott, WBØSKX
Phil Anderson, WØXI

ABSTRACT

The recently-introduced G-TOR protocol for HF data communications employs several features which maintain the throughput of this system in the presence of noise and interference. In this paper we take a closer look at the most important of these features - the hybrid ARQ protocol - in order to provide G-TOR users with a better understanding of the technical details of the protocol and an appreciation of the role of the Golay forward error correcting code in improving the overall system performance. We will demonstrate the advantages of using a hybrid ARQ protocol by presenting a theoretical evaluation of the throughput of the G-TOR hybrid ARQ protocol in the presence of Gaussian noise. Graphs of throughput versus channel bit error rate will show that the combined use of error detection and Golay forward error correction is a powerful approach to extending the throughput of a conventional stop-and-wait ARQ system on the HF bands.

INTRODUCTION

G-TOR (Golay-TOR) was recently introduced by Kantronics as an improved protocol for the structured interchange of digital data between stations operating in the HF Amateur bands. Simplicity was one of the design goals of this protocol - G-TOR was also developed to operate with currently-existing multi-mode TNCs. The system was designed to incorporate many modern digital data processing techniques. For example, Huffman data compression and run length encoding are used together to reduce data redundancy in each transmission. Also, fault tolerant ACKs and NACKs are employed to help prevent needless re-transmissions. The protocol is adaptive in that it allows the TNC to select a data rate based on link quality - 100, 200 or 100 baud - depending upon the number of retrys attempted. However, the most significant feature designed into G-TOR is the use of a hybrid ARQ protocol. The hybrid ARQ protocol employs forward error correction on demand. When channel conditions are good, G-TOR is simply a conventional stop-and-wait (S&W) ARQ system. However, when the channel deteriorates, the Golay forward error correction code is used in such a way that two transmissions of a frame provides enough information for the receiver to have three opportunities to reconstruct an error-free frame. The details of this process are provided in [1]. Also, a more detailed description of G-TOR can be found in [2].

G-TOR is similar in structure to two other popular HF TOR (Teleprinting Over Radio) systems - AMTOR and PacTOR. All of the TOR systems are synchronous half-duplex modes which allow the exchange of digital data between two connected stations. Synchronous operation improves the efficiency of each transmission in that fewer overhead bits are required in each frame to insure bit and frame sync. G-TOR has a longer frame structure than does AMTOR and PacTOR, as shown in Figure 1. In any ARQ system, a short frame is likely to have fewer random errors (and hence fewer rejected frames) than in a long frame. In fact, research has shown that, because of the dynamic nature of the HF communication channel, transmissions should not be much longer than one second in duration. However, every frame must carry overhead bits which are usually independent of the frame length. Therefore, longer frames are more efficient than shorter frames in terms of the information they carry. Obviously, this is an issue which involves an engineering tradeoff for the best compromise between frame efficiency and frame length. G-TOR attacks this problem by making the frame longer to increase frame efficiency, and using interleaving to randomize and distribute the errors which may occur due to burst noise and multipath fading. The G-TOR frame structure is shown in Figure 2, and a comparison of the frame and cycle efficiencies of the HF TOR protocols is provided in Table 1.

THE GOLAY CODE

The real power of G-TOR resides in the properties of the (24,12) extended Golay forward error correcting code and the way it is used in the hybrid ARQ protocol. We decided to use the Golay code for the G-TOR protocol because of its simplicity and its powerful mathematical properties, most notably, the fact that it is a half-

rate code (i.e., the number of parity bits is the same as the number of information bits) and it is invertible. An invertible code is one in which the data bits can be recovered from the parity bits by simply running the parity bits through the Golay encoder. The half-rate feature allows the formation of parity frames which are the same length as data frames. For the interested reader, a useful tutorial on the Golay code is provided in [3].

The power of the Golay code can be seen in the following expression, which essentially gives the probability of one or more errors in a block of n-bits given that the channel bit error rate is ε:

$$\Phi = \sum_{j=0}^{t} \frac{(n)!}{(j)!\,(n-j)!} \varepsilon^{j} (1-\varepsilon)^{n-j} \qquad (1)$$

where Φ is the probability that the (24,12) Golay decoder will not be able to correct all the errors in a block of n-bits (note: n = 24 and k = 12), and t is the error correcting capability of the code The extended Golay code is capable of correcting 3 or fewer errors which may occur in any combination in a 24 bit block, so t = 3 in this case. The impact of using the Golay code is illustrated in Figure 3, which shows the improvement in error performance this code provides over an uncoded system. This is often referred to as the FEC coding *gain.*

Since the Golay code generates 12 parity bits for every 12 data bits, use of the code on every transmission would decrease the overall throughput by a factor of 1/2. This may be an acceptable tradeoff when the signal-to-noise ratio is very low, and the code is needed on every transmission to remove errors; however, in good channel conditions (high SNR) the parity bits would be an unnecessary overhead since they would seldom be needed. The solution to this problem is to use the Golay code only when it is needed, using a hybrid procedure, in which the system operates in a conventional S&W ARQ mode until errors are detected. When errors are detected, a re-transmission request from the information receiving station (IRS) results in the information sending station (ISS) transmitting parity bits instead of information bits. This procedure is summarized in the following section.

THE G-TOR HYBRID ARQ SYSTEM

An important feature of the G-TOR protocol is that it uses a type-II hybrid stop-and-wait ARQ system in combination with the 1/2-rate invertible Golay code for error correction and a 16-bit CRC code for error detection. There are two types of hybrid ARQ [4]; type I ARQ systems send both error correction and error detection parity bits with every transmission, while type II ARQ systems transmit error correction parity bits only when errors are detected in a frame. The error detection code transmitted with each G-TOR frame is a two-byte cyclic redundancy check (CRC) code. The CRC code is used to determine if the frame was received correctly before error correction is initiated; and it is also used after error correction has been completed to insure that the error correction process has successfully removed all errors in the frame.

In the type-II hybrid ARQ system, forward error correction is employed only when it is needed. The advantages of this approach can be illustrated by an example. Before transmitting a block of data to the receiver, the ISS first applies the CRC to the data for error detection, and then encodes the data using the 1/2-rate Golay FEC. The parity bits which are generated by this process are saved at the transmitter and the information bits (including the CRC) are sent. When the IRS receives the data, the check sum is computed. If the check sum passes, the data is accepted, and an ACK is returned to the transmitter. If the check sum fails, a NACK is returned to the transmitter. When the transmitter (ISS) receives an ACK, the parity bits from the first data block are discarded and the next block of data is processed. If a NACK is received, the transmitter sends the block of parity bits which had been held back. When the block of parity bits arrives at the receiver (IRS), the first action taken is to invert the parity bits to obtain data bits. Once data bits are obtained the check sum is computed. If the data (which was obtained by inverting the parity bits) passes the check, it is accepted and sent to the user. If it fails the check, then the parity bits are then combined with the data bits from the first transmission and processed by the FEC decoder. In this way, *two* concecutive transmissions provide the receiver with a total of three opportunities to obtain an error-free block of data.

EVALUATION OF HYBRID ARQ

When evaluating any data communications protocol, the most important parameter is the rate at which information (excluding overhead) is being transferred across the channel to the distant user. This is usually expressed as *throughput efficiency*, which is defined as the ratio of the average number of information bits accepted at the receiver per unit of time to the total number of bits that could be transmitted per unit of time.

Evaluating the throughput efficiency of a hybrid ARQ system is a rather difficult task. In fact, the best way to approach the problem is to

consider both extremes of protocol behavior and develop an upper bound on the performance of the system [5]. Therefore, the approach taken here is to consider two situations - in the first, the protocol is assumed to be a conventional S&W ARQ system, and in the second, the protocol is assumed to use the Golay code on every alternate transmission (essentially a type I ARQ system). The true throughput behavior of the type II hybrid ARQ protocol will then be bounded by the results of both of these computations.

Since G-TOR is a hybrid version of the conventional S&W ARQ procedure, we begin by developing the throughput expression for this case. For reference, the frame timing structure for G-TOR is shown in Figures 1 and 2. Here it is assumed that M-k-bits (where k= 12 and M = # of 12-bit blocks in the frame) are transmitted at r bits/sec in a single frame followed by an interval of T-seconds during which the information receiving station (IRS) is given the opportunity to acknowledge the correct or incorrect receipt of the frame. Note that all M-k-bits in the frame do not carry information. There are a certain number, of bits devoted to overhead - frame status and error detection code parity bits for example. Therefore, we can say that there are a-bits of information in a single frame and (n – a) bits of overhead.

In one complete frame interval, the transmitter could conceivably transmit $(M \cdot k + rT)$ bits of information if it does not stay idle and if all the bits are information bits (i.e., no overhead).. These assumptions allow us to compute the true throughput efficiency of the S&W ARQ system. The throughput efficiency (defined as η), according to our definition can be interpreted as:

$$\eta = \frac{\text{Avg \# info bits received}}{\text{Max \# bits that can be transmitted}} \qquad (2)$$

Where the average number of information bits received is simply $(\alpha \cdot P)$, where P is the probability that the received information will be accepted by the receiver, either because the frame arrived without errors, or because it arrived with a correctable number of errors. The throughput efficiency for the S&W ARQ system is therefore,

$$\eta = \frac{\alpha P}{M k + r T} \qquad (3)$$

Where $P = ((1 - \varepsilon)^k)^M$ with ε the bit error probability of the received data. A more detailed derivation of this expression is provided in [4].

Using (3) we can evaluate the throughput performance of G-TOR as a conventional S&W ARQ system. This will provide a baseline for comparison of the performance improvement with hybrid ARQ. The various parameters used in the evaluation of (3) are provided below:

r (bits/sec)	100 bps	200 bps	300 bps
T (sec)	0.48	0.48	0.48
a (bits)	168	360	552
M·k (bits)	192	384	576
M (blocks)	16	32	48

The results of these computations are plotted in Figure 4, expressed *as throughput* which is defined in terms of througput efficiency as,

$$\text{Throughput (bits/sec)} = \eta \cdot r \qquad (4)$$

Now let's include the effect of the Golay code. The Golay code significantly improves the error performance of the overall system by using the parity bits to correct up to 3 errors in a single 24-bit block. of received data. If we assume that the channel is poor and transmissions alternate between data and parity, we have established the condition for the lower bound on throughput for the system. By defining $P = \Phi^M$ in (3) and dividing by 1/2 to account for the repeated transmission, the performance of the hybrid ARQ system can be observed by plotting the throughput versus the channel bit error rate, using (3) and (4). The result is shown in Figure 5.

INTERPRETATION OF RESULTS

The essential performance of the G-TOR hybrid ARQ system can be appreciated by examining Figure 5, which evaluates the protocol at the 200 bits/sec data rate. In this graph we see that the actual information throughput (excluding overhead) is 150 bits/sec when the channel is good. As long as the received signal is strong and no interference is present, G-TOR is functioning in the S&W ARQ mode. At the opposite extreme, when the conditions are bad, G-TOR is alternating data frames and parity frames as errors are occuring in every transmission. During this time, the throughput has been cut in half because of the constant need for the parity bits.

The Golay code begins to be used frequently when the error rate is above 10^{-3}. When the channel error rate reaches approximately 2×10^{-3} the hybrid ARQ system extends the throughput and keeps it constant as the channel continues to deteriorate. The system performance is effectively extented by a factor of 10 - a considerable improvement.

CONCLUSIONS

In this paper we have briefly examined in simple form, the theoretical throughput performance of the G-TOR hybrid ARQ protocol. The performance of the system was derived by quantifying the system behavior under both poor channel and good channel conditions. In all cases we have assumed that Gaussian noise is the only interfering signal. A more realistic evaluation would need to take into account the presence of burst errors caused by manmade and natural phenomena. This type of evaulation is more reasoably accomplished through simulation or through exhaustive on-air testing. The results provided here verify that the Golay code, when used in a type II hybrid ARQ system, is an effective deterent to random bit errors.

REFERENCES

[1] G. Prescott and P. Anderson, "Hybrid ARQ for HF Data Transmission: Forward Error Correction on Demand," *Communications Quarterly*, to appear in August 1994 issue.

[2] G. Prescott and P. Anderson, "G-TOR: A Hybrid ARQ Protocol for Narrow Bandwidth HF Data Communication, *QEX*, May 1994, pp. 12- 19.

[3] J. Iovine, "Using the Golay Error Detection and Correction *Code," The Computer Applications Journal*, Issue #48, July 1994, pp. 24 - 35.

[4] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs NJ, 1983.

[5] S. Lin and P Yu, "A Hybrid ARQ Scheme with Parity Retransmission for Error Control of Satellite Channels," *IEEE Transactions on Communications*, Vol. COM-30, No. 7, July 1982, pp. 1701 - 1719.

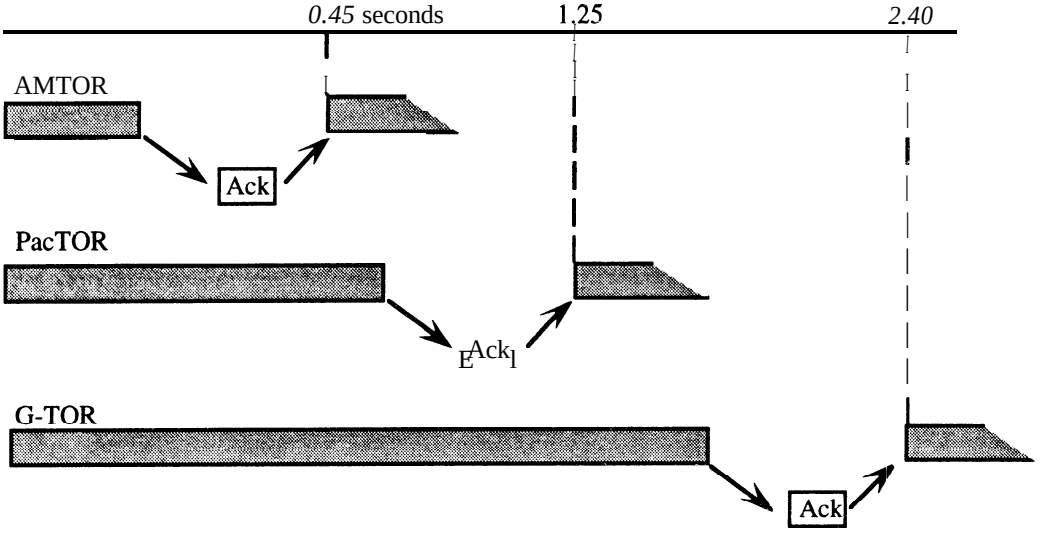
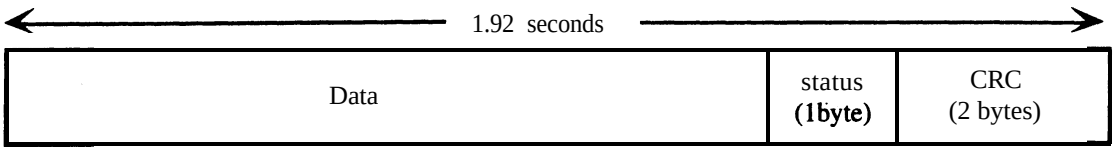


Figure 1 - TOR ARQ System Timing



69 data bytes @ 300 baud
45 data bytes @ 200 baud
21 data bytes @ 100 baud

Figure 2 - G-TOR Frame Structure before Interleaving

Table 1 - HF ARQ Protocols

Mode	Baud Rate	Data (bits/frame)	Frame efficiency	Cycle efficiency	Ideal bits/sec
AMTOR	100	15	71.4%	33.33%	33.3
PacTOR-LP	100	64	66.7%	45.71%	45.7
PacTOR-LP	200	160	83.3%	57.14%	114.3
PacTOR	100	64	66.7%	5 1.20%	51.2
PacTOR	200	160	83.3%	64.00%	128.0
G-TOR	loo	168	87.5%	70.00%	70.0
G-TOR	200	360	93.8%	75.00%	150.0
G-TOR	300	552	95.8%	76.67%	230.0

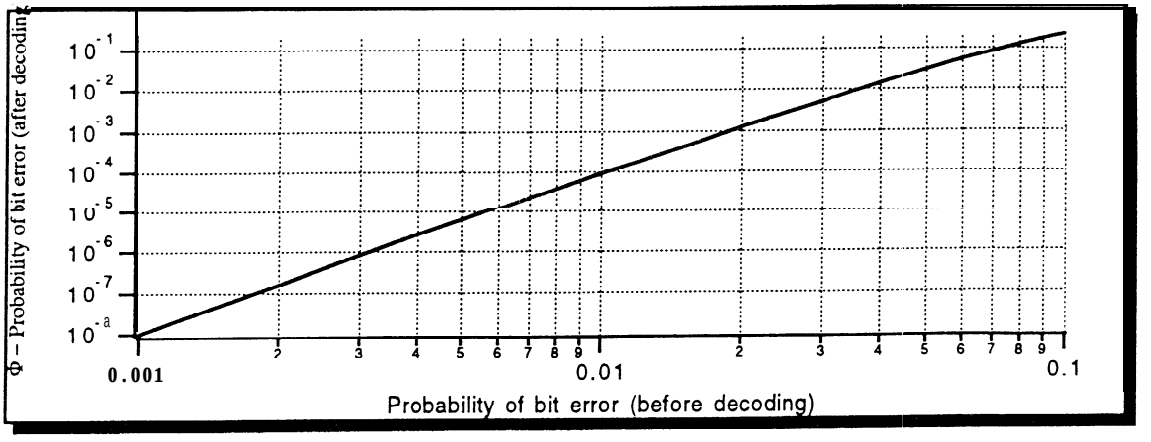


Figure 3 - Performance improvement provided by the (24,12) extended Golay code

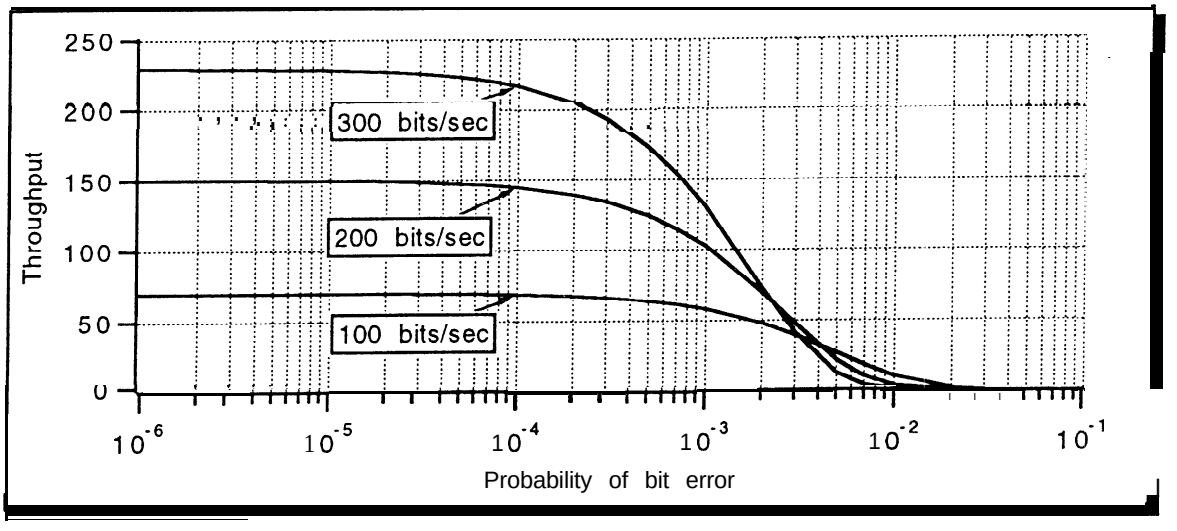


Figure 4 - S&W **ARQ** Performance of G-TOR (without the use of FEC)

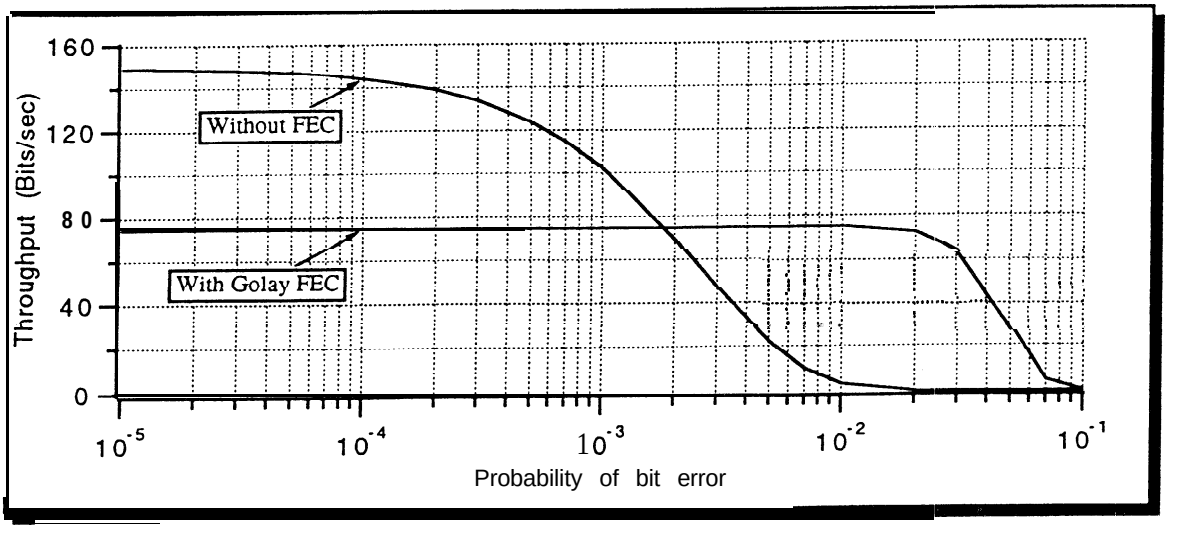


Figure 5 - Performance of G-TOR with hybrid ARQ at 200 bits/sec

ON FRACTAL COMPRESSION OF IMAGES FOR NARROWBAND CHANNELS AND STORAGE

W. Kinsner, VE4WK

Department of Electrical and Computer Engineering
and Telecommunications Research Laboratories
University of Manitoba
Winnipeg, Manitoba, Canada R3T 5V6
E-mail: kinsner@ee.umanitoba.ca; Radio: VE4WK@VE4KV.#WPG.MB.CAN.NA

Abstract

Fast transmission of digital images and fast video over packet radio can only be possible by compression of the original uncompressed source material that contains many bits. This paper provides several comments on a new class of compression techniques based on fractals. This approach may exceed by far the compression limit of the JPEG technique.

1. INTRODUCTION

Living with ASCII text on packet is no longer satisfactory for me. I would like not only to see a photograph of a radio pal, as well as colour weatherfax, aerial photographs, scientific and technical visual representations, and even fast video, but also to *hear* speech and various sounds, including music. This applies equally well to the fast universal network of networks, the Internet, because its traffic is increasing exponentially. It will also apply to the future information superhighway. Thus, there is an increasing fundamental need for transmission and storage of still and moving images through the available communications channels. If the transmission occurs over radio, the bandwidth of the channels is always narrow because the spectrum is finite. Consequently, the number of bits in the source material (i.e., uncompressed pictures) must be reduced in order to reduce the transmission time of the picture, or send more pictures over the same channel within the same time.

There are numerous lossless and lossy methods and techniques capable of compressing images, as shown in Fig. 1 [Kins91a], [Kins9 1 b]. *The lossless compression* approach is concerned with the removal of *redundancy* from the source (i.e., bits that carry no additional information). The approach is called lossless because no information is lost in the compression and reconstruction process, although fewer bits are transmitted. Examples of such compression implementations are the Huffman [Huff52] and Lempel-Ziv-type [Welc84] and arithmetic coding [WaFK93], [WiNC87] techniques [DuKi91], as found in the PKZIP and other compression computer programs used for archiving and packet transmission.

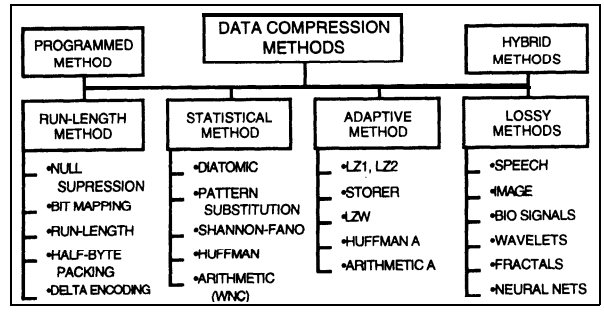


Fig. 1. Compression methods and techniques.

In contrast, the lossy compression approach removes *irrelevancy* from the source (i.e., the features that are not important in human perception). This approach leads to some losses, though imperceptible, in the reconstructed image. Examples of such techniques are the JPEG (Joint Photographic Experts Group) [PeMi93] and wavelet compression [LaKi94]. The wavelet compression reference describes our work on the very promising class of techniques that are ideally suited for perceptually-driven

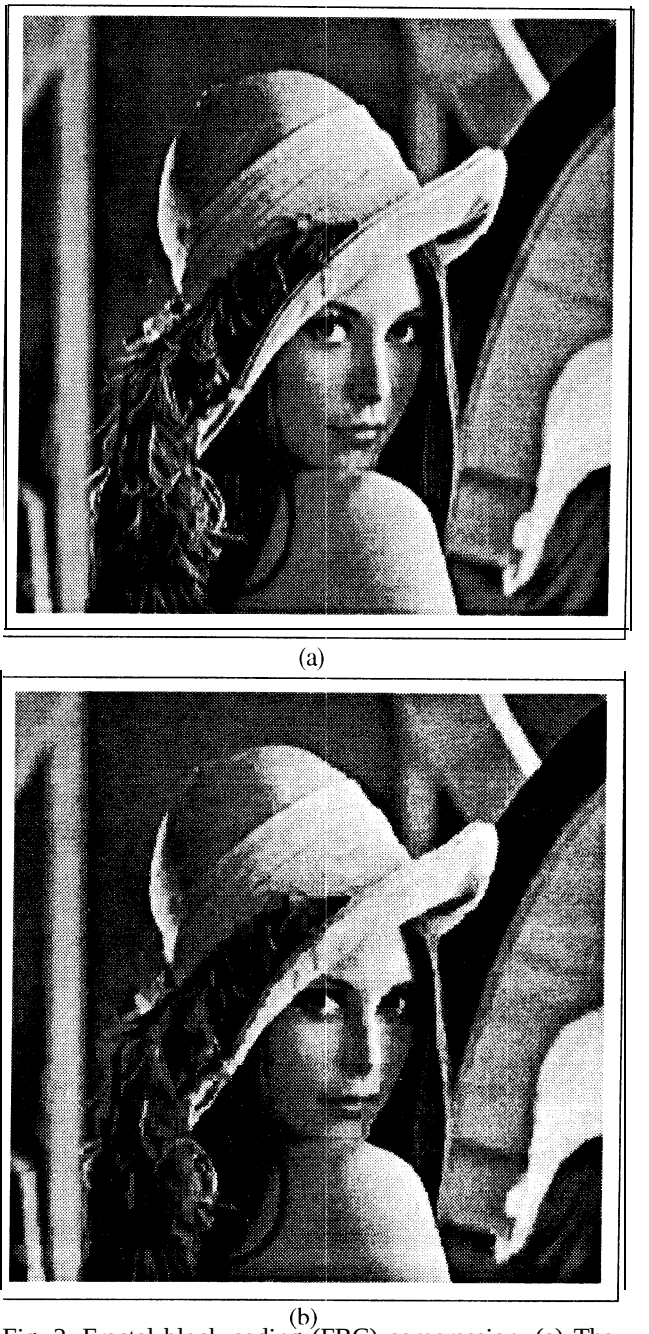
compression. The JPEG reference provides the most accurate description of the international standard for **digital** compression of continuous-tone still images. It **defines a** toolkit of processes for lossy and lossless encoding and decoding of images. It also describes image-coding techniques, with emphasis on Huffman coding, arithmetic coding, predictive coding, QM coding and the discrete cosine transform (DCT) [ChSF77], as well as the JPEG - modes of operation, signalling conventions, and structure of compressed data. A brief overview is also provided of both the JBIG (Joint Bi-Level Image Experts Group), and MPEG (Moving Picture Experts Group).

Barnsley [Barn88] and Jacquin [Jacq90] introduced compression techniques based on fractals [PeJS92] that far exceed the limit of the JPEG scheme, and can reach compression ratios of hundreds or even thousands to one. We have introduced a reduced-search fractal block coding technique, with smaller compression ratios at this time, but much shorter computing times and a systematic analysis of images [WaKi93a].

The lossy techniques can be combined with the lossless techniques in order to remove any redundancy left in the compressed code. Such concatenated codes are very compact, as demonstrated by combining a fractal code with the arithmetic code [WaKi93b]. Furthermore, forward error correction may also bc introduced to protect the code against errors during transmission, as described by this author [Kins90].

Is compression the only problem for data and signal transmission and storage? The answer is yes, if we consider the number of bits delivered to the end-user as the only objective of the solution. However, if we deliver more and more bits, how will the user navigate through them? In fact, if there are more bits to chose from for transmission, how will the user know which bits should be transmitted? In his keynote address at a recent conference in Halifax, NS, the director of the Media Lab at the Massachusetts Institute of Technology (MIT) stated that "people don't want *more* bits, they want the *right* bits". So, the additional problem is how to select (filter) the right bits. Searchability is associated with such filtering of information.

While text can be searched for keywords easily, searching of mathematical expressions, tabular data, and musical scores is much more difficult. Even more difficult is searching of images and sound. This problem will have to be resolved, if we want to have the multimedia form of transmitted information.



(a) (b)

Fig. 2. Fractal block coding (FBC) compression. (a) The original image of Lena with 256x256 pixels and 8 bpp. (b) Reconstructed image from FBC with compression of 14.3: 1 at 0.56 bpp and 29.1 dB.

3. FRACTAL COMPRESSION

Fractal data compression is an alternative to JPEG and vector quantization, with a much higher compression potential. It has attracted a great deal of interest since Barnsley's introduction of *iterated functions systems* (IFS), a scheme for compactly representing intricate image structures [Barn88], [BaHu93]. Although the applicability of IFSs to the compression of complicated color or gray-scale images is hotly debated, other researchers have applied the concept of self-similarity (fundamental to fractals) to image compression with promising results.

We have developed a block-oriented fractal coding technique for still images [WaKi93], [Wall93] based on the work of Arnaud Jacquin [Jacq90]. Jacquin's technique has a high order of computational complexity, $O(n^4)$. We have used a neural network paradigm known as *frequency sensitive competitive learning* (FSCL) [AKCM90] to assist the encoder in locating fractal self-similarity within a source image. The approach identifies all the affine transformations within the image without the assistance from a human operator.

A judicious development of the proper neural network size for optimal time performance has been provided. Such an optimally-chosen network has the effect of reducing the time complexity of Jacquin's original encoding algorithm from $O(n^4)$ to $O(n^3)$. In addition, an efficient distance measure for comparing two image segments independent of mean pixel brightness and variance is developed. This measure, not provided by Jacquin is essential for determining the fractal block transformations prescribed by Jacquin's technique.

Figure 2 shows the results from our fractal technique. Figure 2a is the original image of Lena represented by 256x256x8 bits. Figure 2b shows Lena reconstructed from the compressed form (14.3: 1 or 0.56 bits per pixel) by the FBC. The peak signal to noise ratio (PSNR) is 29.1 dB. The image quality is quite high, with some details segmented into slightly visible blocks. The block nature of the image is much reduced if we take a higher resolution image (e.g., 512x5 12x8). An added unique feature of FBC is its ability to zoom on details (not shown here).

4. FEATURE EXTRACTIONFOR SEARCHABILITY

The fractal compression techniques just described, together with a learned vector quantization [FeLK93], lends itself to searching in that it provides codebooks of characteristic features.

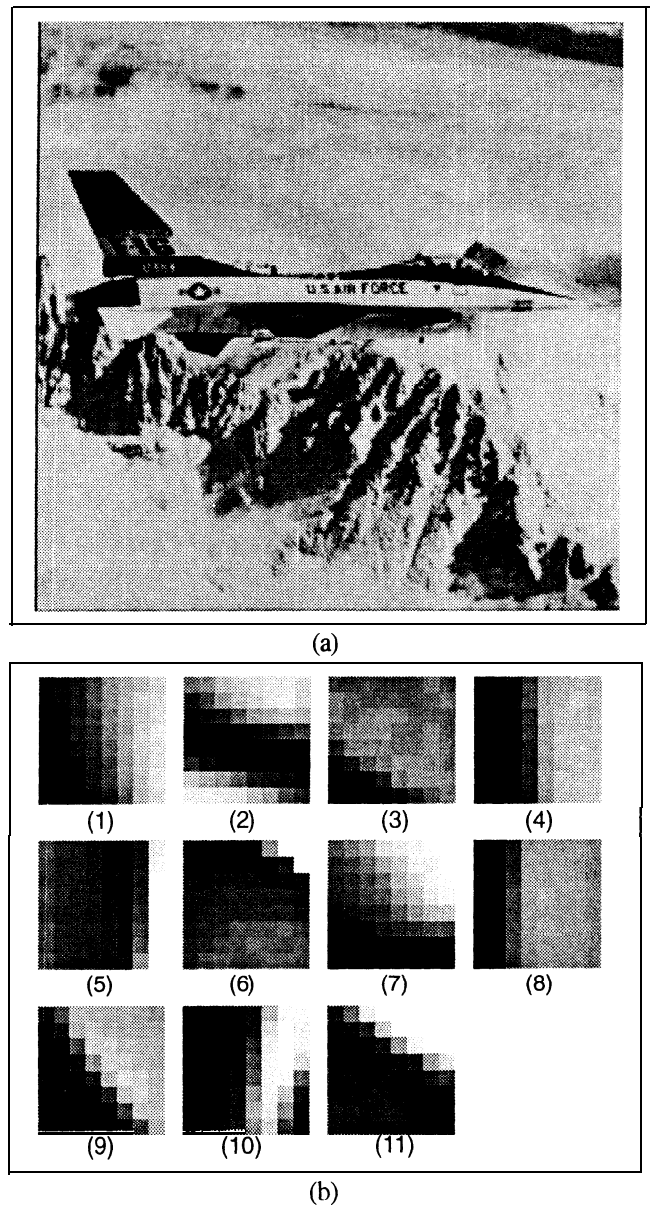


Fig. 3. Finding characteristic features in the form of a codebook for searchability. (a) The original image of AIRPLANE with 256x256 pixels and 8 bpp. (b) Optimal codebook for the image.

The codebooks are extracted in an adaptive fashion, without a "teacher" showing what is and what is not significant in the image or a temporal signal, It is then proposed to search for characteristic features in such objects using the codebooks rather than the direct digital representation of the objects.

Figure 3 shows an example of an optimal codebook (Fig. 3b) for the scene containing an airplane (Fig. 3a). The codebook has been obtained using our fractal compression. To achieve searchability, the codebook must be correlated to and associated with the objects in the image. Work on the association scheme continues.

5. DISCUSSION

As any users of the Internet and the future information superhighway, the users of packet radio will require diverse material ranging from simple text, formatted text, mathematical expressions, tabular data, line drawings, maps, gray-scale or colour still images and photographs, phonetic transcription of speech, music scores, and other symbolic representations, to hypertext, animation and video, actual recordings of sounds such as electrocardiograms or other biomedical signals, telephone and broadcast-quality speech, and wideband audio. Such representations lead to large files.

Consequently, efficient storage and transmission requires data and signal compression. Since the lossless arithmetic coding is better than other statistical techniques, it can be used for critical data where no bit can be lost.

In contrast, lossy compression techniques based on wavelets, neural networks, and particularly fractals have good prospects of becoming important in such a delivery of information. These techniques also provide tools for the development of universal feature codebooks that can be used for searching of nontextual material.

ACKNOWLEDGEMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council, Manitoba Telephone System, and now the Telecommunications Research Laboratories (TRLabs).

REFERENCES

[AKCM90] S.C. Ahalt, A.K. Krishnamurthy, P.Chen, and D.E. Melton, "Competitive learning algorithms for vec tor quantization," *Neural Networks,* Vol. 3, No. 3, pp. 277-290, 1990.

[Barn88] M.F. Bamsley, *Fractals Everywhere. New* York (NY): Academic, 1988, 396 pp.

[BaHu93] M.F. Bamsley and L.P. Hurd, *Fractal Image Compression.* Wellesley (MA): AK Peters, 1993, 244 pp.

[ChSF77] W.H. Chen, C.H. Smith, and S.C. Fralick, "A fast computational algorithm for the discrete cosine transform," *ZEEE Trans. Communications,* Vol. 25, No. 9, pp. 1004-1009, September '1977.

[DuKi9 I] D. Dueck and W. Kinsner, "Experimental study of Shannon-Fano, Huffman, Lempel-Ziv-Welch and other lossless algorithms," *Proc. 10th Computer Networking Con.,* (San Jose, CA; Sept. 29-30, 1991), pp. 23-3 1, 1991.

[FeLK93] K. Ferens, W. Lehn, and W. Kinsner, "Image compression using learned vector quantization," *Proc. IEEE Communications, Computers & Power Conf.*, WESCANEX'93 (Saskatoon; SK; May 17-18, 1993), IEEE Cat. No. 93CH3317-5; pp. 299-312, 1993.

[Huff52] D.A. Huffman, "A method for constructing minimum-redundancy codes," *Proc. IRE,* Vol. 40, pp. 1098-l 101, September 1952.

[Jacq92] A.E. Jacquin, "Image: coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing,* Vol. 1, No. 1, pp. 18-30, January 1992.

[Kins90] W. Kinsner, "Forward error correction for imperfect data in packet radio," *Proc. 9th Computer Networking Conf.* ; (London, ON; Sept. 22, 1990), 1990, pp. 141-149.

[Kins91a] W. Kinsner, "Review of data compression methods, including Shannon-Fano, Huffman, arithmetic, Storer, Lempel-Ziv-Welch, fractal, neural network, and wavelet algorithms,"

Technical Report, DEL9 1- 1, Winnipeg, MB: Dept. of Electrical and Computer Engineering, University of Manitoba, 157 pp., January 1991.

[Kins91b] W. Kinsner, "Lossless data compression algorithms for packet radio," *Proc. 10th Computer Networking Conf.*; (San Jose, CA; Sept. 27-29, 1991), 1991, pp. 67-75.

[LaKi94] A. Langi and W. Kinsner, "Wavelet compression for narrowband channels," *Proc. I994 Digital Communications Conf.;* (Bloomington MN; August 19-21, 1994), 1994, this volume.

[PeJS92] Heinz-Otto Peitgen, Hartmut Jürgens and Dietmar Saupe. *Chaos and Fractals: New Frontiers of Science. New* York (NY): Springer-Verlag. 1992,984 pp.

[PeMi93] William B. Pennebaker and Joan L. Mitchell. *JPEG: Still Image Data Compression Standard.* New York (NY): Van Nostrand Reinhold. 1993, 638 pp.

[WaFK93] L. Wall, K. Ferens, and W. Kinsner, "Real-time dynamic arithmetic coding for low-bit-rate channels," *Proc. ZEEE Communications, Computers & Power Con.,* WESCANEX'93 (Saskatoon; SK; May 17-18, 1993), IEEE Cat. No. 93CH3317-5; pp. 381-391, 1993.

[WaKi93a] L.Wall and W. Kinsner, "A fractal block coding technique employing frequency sensitive competitive learning," *Proc. IEEE Communications, Computers & Power Conf.*; Wescanex'93 (Saskatoon, SK; May 17- 18, 1993), IEEE Cat. no. 93CH33 17-5, pp. 320--329, 1993.

[WaKi93b] L. Wall and W. Kinsner, "Reduced search fractal block coding concatenated with entropy coding," *9th International Conference on Mathematical and Computer Modelling Record,* ICMCM'93 (Berkeley, CA; July 26-29, 1993) (in print).

[Wall93] L. Wall, "Reduced search fractal block coding using frequency sensitive neural networks," *M.Sc. Thesis,* Department of Electrical and Computer Engineering, University of Manitoba, 200 pp., May 1993.

[Welc84] T.A. Welch, "A technique for high-performance data compression," *IEEE Computer,* Vol. 17, pp. 8-19, June 1984.

[WiNC87] I.H. Witten, R.M. Neal, J.G. Cleary, "Arithmetic coding for data compression," *Comm. ACM,* Vol. 30, No. 6, pp. 520-540, June 1987.

FAST CELP ALGORITHM AND IMPLEMENTATION FOR SPEECH COMPRESSION

A. Langi, VE4ARM, W. Grieder, VE4WSG, and W. Kinsner, VE4WK

Department of Electrical and Computer Engineering
and Telecommunications Research Laboratories
University of Manitoba
Winnipeg, Manitoba, Canada R3T 5V6
Tel.: (204) 474-6992; Fax: (204) 2750261
eMail: kinsner@ee.umanitoba.ca

ABSTRACT

This paper describes a fast algorithm and implementation of *code excited linear predictive* (CELP) speech coding. It presents principles of the algorithm, including (i) fast conversion of *line spectrum pair* parameters to linear predictive coding parameters, and (ii) fast searches of the parameters of *adaptive* and *stochastic codebooks.* The algorithm can be readily used for speech compression applications, such as on (i) high quality low-bit rate speech transmission in point-to-point or store-and-forward (network based) mode, and (ii) efficient speech storage in speech recording or multimedia databases. The implementation performs in real-time and near real-time on various platforms, including an IBM-PC AT equipped with a TMS320C30 module, an IBM PC 486, a SUN Sparcstation 2, a SUN Sparcstation 5, and an IBM Power PC (Power 590).

1. INTRODUCTION

1.1. Why is CELP Useful ?

Obtaining efficient representation of speech at low bit rates for communication or storage has been a problem of considerable importance, because of technical as well as economical requirements. Telephone-quality digital speech in a *pulse code modulation* (PCM) form requires a 64 kbits/s rate which cannot be transmitted in real time through 6 kHz and 30 kHz channel capacities of HF and VHF bands, respectively. Voice mail and multimedia employ speech storage, demanding efficient ways of storing speech, since one minute of PCM speech already requires 480 kbytes of storage space. Even if the channel can accommodate real-time speech, speech compression allows more communication connections to share the precious channel. Similarly, speech compression allows more speech messages to be stored in the storage of the same size.

This paper describes a speech compression technique for those purposes, called *code-excited linear predictive* (CELP) coding [Atal86] [JaJS93], which obtains bit rates of as low as 4.8 kbits/s, giving a compression ratio of up to 13: 1 [CaTW90]. Although this rate is higher than a 2.4 kbits/s *linear predictive coding* (LPC), speech compressed by CELP has quality, naturalness, and speaker recognizability, which are missing from the LPC.

The importance of CELP goes beyond its quality vs. bit-rate performance, as it *provides a generic structure for future generation of' *perceptual* speech coders [JaJS93]. All speech compression techniques have been based on two intrinsic operations: removal of *redundancy* and removal of *irrelevancy. The* first operation uses *prediction* and/or *transforms* to remove redundant data, thus reducing the bit rates. The second operation further reduces the bit rates through quantization of(i) the time components of the prediction error or (ii) the transform coefficients, allowing mathematically non-zero but *imperceptible* reconstruction error or distortion.

If further compression is still required, the coder minimizes the error perceptibility by exploiting *masking* properties of human speech perception. To certain extent, the speech energy itself perceptually masks the distortion. Thus the same energy levels of distortion have different perceptual effect if applied to speech signals with different energy levels. This approach promises a new level of higher quality and lower bit rate speech compression [JaJS93]. Coders that minimize perceptual distortion (such as CELP) are called perceptual coders.

One novelty of CELP is in incorporating the masking property in a working, practical scheme. Such incorporation is non trivial because perceptual distortion measures lack tractable means that have often been available in the traditional distortion energy measure.

The CELP solution to this problem is by using an analysis-by-synthesis approach, where the perceptual distortion is literally measured. CELP then exploits the computational structure, resulting in a sophisticated, practical compression technique. Clearly, the computational cost is very high.

1.2. Conceptual CELP

As shown in Fig. 1, a conceptual CELP structure [ScAt85] consists of:

a. two predictors (pitch and spectral predictor filters) to remove redundancy caused by long and short term correlations among speech samples, respectively; and

b. a close-loop, perceptual vector quantizer utilizing a codebook to remove irrelevancy indirectly from the time components of the prediction error.

The codebook stores random (stochastic) signals as prototypes of excitation signals for the two predictor filters. Furthermore, a perceptual weighting filter ensures that mean-square error measurement reflects the perceptual error measurement.

The CELP compressed speech then consists of:

a. a set of spectral predictor parameters;

b. a set of pitch predictor parameters; and

c. codebook (entry and gain) parameters.

It is these CELP parameters than can be transmitted or stored at rates as low as 4.8 kbits/s.

The speech compression algorithm begins by obtaining the predictor parameters, and then searching for codebook parameters corresponding to excitation prototype that minimizes the perceptual error. The CELP decompressor uses the codebook parameters to produce the excitation signal, exciting the cascade of pitch and spectral filters, resulting with the decompressed speech.

The selection of the predictors and the quantizer is by no means arbitrary. They match elements of a model of human speech production system [Lang92]. The model consists of an excitation source and a vocal tract. During voiced speech articulations, the excitation source produces quasi periodic pulses which excite the vocal tract. The pulses are subjected to resonance and anti-resonance processes in the vocal tract according to the changes in the vocal tract shape over time, resulting in audible and meaningful speech. Similar processes take place during *stop* and fricative articulations. However, the excitation source should produce noise-like excitations instead. In matching the model, CELP uses the spectral predictor filter to perform vocal tract function. The pitch predictor filter (usually a one-tap, all-pole filter) ensures the quasi-periodicity of the spectral filter excitation. In this cascaded filter structure, it is known that voiced speech signals have excitations of Gaussian distribution. Thus the codebook members represent such excitations. It also accommodates excitation for stops and fricatives. The fact that the CELP structure serves both signal compression principles (i.e., redundancy and irrelevancy removals) and speech production model (i.e., an articulation source and vocal tract) is the reason for the CELP highly successful performance.

1.3. Implementation Problem

Despite its concept maturity, real-time CELP implementation is still a complex problem. The codebook searching is so computationally demanding that a direct implementation requires very long computation time, much more than real-time requirement. In the searching process, each prototype

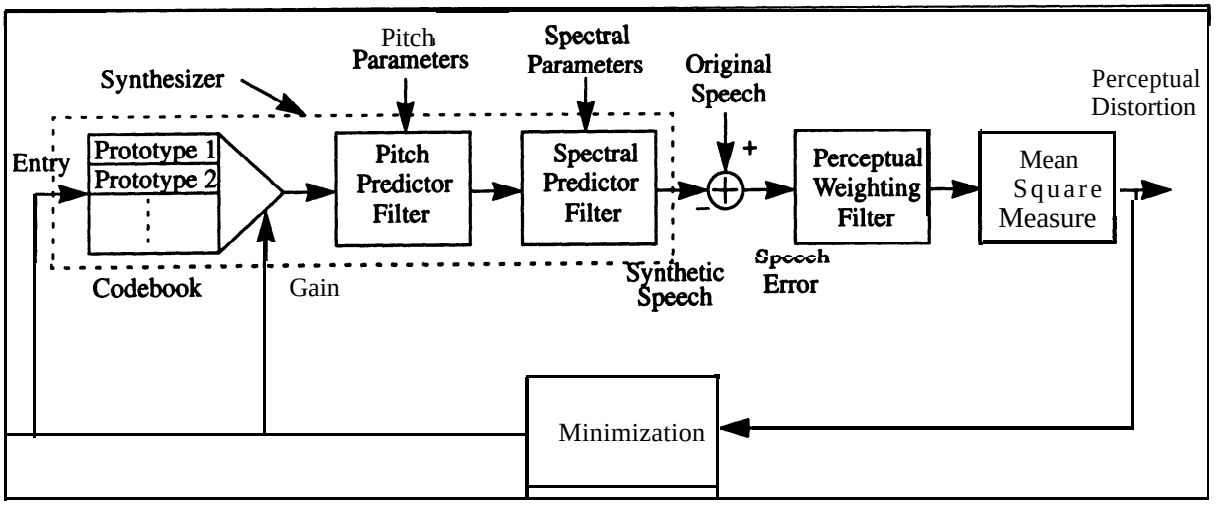


Fig. 1. Conceptual CELP analyzer.

must go through three filtering (the pitch, spectral, and perceptual filters) and one mean-square processes. It is easy to show that a brute-force approach would require a processor with more than 34 million MIPS, for a real-time CELP [Lang92]. An early 'practical' CELP implementation required 125s of Cray- 1 computation time to process one second speech [ScAt85], while real-time procedure must process one second of speech in one second or less.

Thus, a practical CELP system must employ fast algorithms, which exploit the computational structure of a CELP scheme. In the process of developing practical CELP, the actual structure becomes significantly different from the conceptual one, while still performing the same functions (see [Lang92] for details on the transition). For example, the spectral parameters are quantized and represented now by a set of *line-spectrum pairs* (LSP) [SoJu84]. The pitch filter becomes another codebook, called *adaptive code book* (ACB). The codebook of the random signals is then called *stochastic code book* (SCB).

Unfortunately, the fast algorithm has significantly increased the implementation complexity as the optimization blurs the structure in favor of speed. The algorithm now combines the spectral predictor and the perceptual weighting filter into one filter. A joint optimization scheme searches for the suboptimal combination of codebook parameters, instead optimal combination through total exhaustive search of all combinations, as implied by the conceptual structure. The use of a special SCB results in a fast iterative search, in which the results of the perceptual distortion calculation from current prototype helps the calculation of that of the next prototype. It should be noted that although there is a proposed U.S. Federal Standard (FS) 1016 CELP [CaTW90] which describes each bit in the compressed speech, it does not specify how to obtain the compressed speech, leaving it to CELP implementors to develop one.

1.4. Paper Overview

The remaining part of this paper describes a practical, near real-time CELP algorithm, which reduces the computational power requirement by a factor of more than 175,000. Section 2 describes the procedures to compress and decompress speech. This paper focuses mainly on the description of algorithms compatible with the FS-1016 to enable communication with other FS-1016 systems. In Section 3, we briefly explain the actual computer implementation, resulting in performance ranging from 14 to 0.85 of real time, depending on the platform. The algorithm has been implemented on an IBM PC-AT equipped with a TMS320C30 (C30) evaluation module (EVM) [LaKi91],[Lang92]. The system is suitable for PC-based packet radio or speech recording systems. The algorithm has also been ported to the various UNIX platforms as well as MS Windows 3.1 platform for a voice mail development. Section 4 discusses performance of the various implementations, including their limitations. Finally, Section 5 provides conclusions.

2. FAST CELP PROCEDURES

2.1. Input and Output

In practice, CELP is a block coding, in which a *frame* of 240 PCM speech samples $s[n]$ (with a total of 1.92 kbits) denoted as a vector s is converted to 144 bits of compressed data, called FS-l016 CELP parameters or *data stream.* The CELP parameters now consist of:

a. the *line spectrum pair* (LSP) parameters;
b. the *adaptive codebook* (ACB) parameters; and
c. the *stochastic codebook* (SCB) parameters.

All LSP, ACB, and SCB parameters are entries (indexes) of quantization tables and codebooks, namely LSP table, ACB, ACB gain table, SCB, and SCB gain table [LaKi90], [LaKi91]. They all require 138 bits only. The remaining 6 bits can be used for error correction, synchronization, and future expansion.

Naturally, the CELP procedures should perform a CELP compressor and decompressor system extracting CELP parameters from $s[n]$, and reconstructing s back from the FS-1016 data stream. Specifically, a CELP compressor (usually called *analyzer)* requires (i) LSP analysis procedure to obtain the LSP parameters, and (ii) codebook search procedure for both ACB and SCB parameters, while a CELP decompressor (usually called *synthesizer)* requires speech synthesis procedure. We describe the procedures as follow.

2.2. LSP Analysis

The CELP analyzer obtains the LSP parameters through the following three steps: (i) performing *linear predictive coding* (LPC) analysis on the PCM samples to represent spectral information [Pars86], [Proa83], (ii) converting the LPC parameters into LSP parameters [KaRa86], [HaHe90] for efficient representation, and (iii) ensuring LSP parameter stability.

2.2.1. LPC Analysis

The aim of LPC analysis is to obtain LPC parameters a_i (collectively denoted as a) corresponding to the spectral filter. The spectral (or LPC) filter models a human vocal tract. One most common model is a 10-order all-pole digital filter $H(z)$ with ten coefficients a_i, as follows

$$H(z) = \frac{1}{1 + \sum_{i=1}^{10} a_i z^{-i}} = \frac{1}{A(z)} \quad (1)$$

Let the input (excitation) of this filter be a zero-mean signal **t.** The output of this filter is then $\hat{\mathbf{s}}$, according to (in z-domain notations)

$$\hat{S}(z) = H(z)\,T(z). \quad (2)$$

For a given s, the LPC analysis finds a that minimizes $\|\mathbf{s} - \hat{\mathbf{s}}\|$. The elements (a_i) of such a vector a are LPC parameters, which are the solutions of a linear equation system

$$0 = \sum_{i=O}^{10} a_i r_{i\ j} \qquad a_0 = 1 \qquad j = 1, \ldots 10 \quad (3)$$

where r_i are autocorrelation terms defined as

$$r_i = \sum_{n=i}^{N-1} s[n]\,s[n-i] \qquad i = 0, \ldots 10 \quad (4)$$

2.2.2. LPC to LSP Parameter Conversion

The system must quantize a using the LSP analysis since a_i are 10 real numbers which require too many bits (10×16=160 bits) for representation. On the other hand, the LSP parameters (we call them LSP_j) **are** more efficient (only 34 bits) because they are ten integers ranging form 0 to 8 (or to **16),** corresponding to the entries of a suitable LSP table.

To show the conversion, we first show that a can ***be*** represented by z_i, which are zeros of two polynomi**als** *p(z)* and $q(z)$ related through

$$\begin{aligned} A(z) &= \frac{1}{2}(p(z) + q(z)) \\ p(z) &= A(z) + z^{-11}A(z^{-1}) \\ q(z) &= A(z) - z^{-11}A(z^{-1}) \end{aligned} \quad (5)$$

Clearly, polynomials $p(z)$ and $q(z)$ represent $H(z)$. In other words, zeros z_i of $p(z)$ and $q(z)$ (eleven each) can represent a.

Furthermore, z_i can be represented fully by ω_i, as

$$\omega_i = \arg(z_i); \qquad i = 0, \ldots, 9 \quad (6)$$

where **arg(•)** is ***the*** argument of a complex variable. The proof relies on ***the*** fact that $z = 1$ **and** $z = -1$ are always the zeros of $p(z)$ and $q(z)$, respectively. Thus the 20 remaining zeros are sufficient to represent $p(z)$ and $q(z)$. Furthermore, all z_i are symmetric about the real axis, and lie on the unit circle in the z-plane. Thus, 10 zeros (below the real line) are actually redundant, leaving us with the remaining 10 significant zeros, which uniquely corresponding to 10 values of ω_i through **Eq.** (6). Furthermore, it can be shown that ω_i with even and odd i correspond **to** $p(z)$ and $q(z)$, respectively. We then conclude that these 10 values of ω_j can reconstruct all the zeros of p(z) and $q(z)$, ***thus*** representing a. Equivalently, for a given **a, we** *can al*ways derive such ω_j.

Having obtained ω_j, we can efficiently represent them through quantization. Although we can directly quantize a_j, the dynamic range of a_j is high (i.e. there are many significant values of a_j), requiring many quantization steps to achieve low quantization error. On the other hand, each ω_j has a much limited dynamic range, since the ranges of ω_j are disjoint subintervals S_j, in a real-number interval of 0 to π, i.e.,

$$\begin{aligned} &\omega_j \in S_j; \qquad 0 \le \bigcup_j S_j < \pi; \\ &i \ne j \Rightarrow (S_j \cap S_i = \varnothing); \qquad j = 0, \bullet \quad **,9 \end{aligned} \quad (7)$$

Thus, fewer quantization steps for ω_j can achieve the same quantization error.

We then use the FS-1016 LSP table to quantize ω_j For each ω_j, FS-1016 sets a ***list*** of 8 possible quantized values of ω_j (or 16 if j is 2 to **5),** covering S_j and its neighborhood. Thus, there are 10 lists, namely list j, j = 0 to 9, collectively called the FS-1016 ***LSP table.*** Let ***LSPTable[j,i]*** be a particular quantized value indexed by ***i*** in list j, where ***i*** is from 0 to 7, or to 15. We quantize ω_j by selecting ***i*** such that ***LSPTable[j,i]*** is the closest value to ω_j in list j. **Now,** assigning such an ***i*** to LSP_j and performing similar steps for all j, we have LSP_j ***as*** a representation of the quantized ω_j. We have called those LSP_j as LSP parameters, which can now represent a. This representation is efficient because we only need **3+4+4+4+4+3+3+3+3+3** = 34 bits for each a, instead of 160 bits in the original floating-point form.

One advantage of using the FS-1016 LSP table is that we can derive a fast LSP conversion algorithm, by searching the table without actually knowing the exact zeros. There are numerical methods such as **Newton-**Rhapson and Jenkins-Traub **[PTVF92]** for finding the zeros of $p(z)$ and $q(z)$, but they are tedious. Furthermore, the exact ω_j must later be quantized anyway.

A different and faster approach is by checking zero-crossing of a new pair of polynomials $\tilde{p}(x)$ and $\tilde{q}(x)$. These polynomials are related to *p(z)* and *q(z)* in the fact that their zeros, x_i, are

$$x_i = \cos \omega_i \tag{8}$$

Such $\tilde{p}(x)$ and $\tilde{q}(x)$ must then take a form of

$$\tilde{p}(x) = \sum_{i=O}^{5} b_i x^i$$
$$\tilde{q}(x) = \sum_{i=O}^{5} c_i x^i \tag{9}$$

where the coefficients **b** and c are

$$\begin{aligned} b_5 &= 32 \\ b_4 &= 16p_1 \\ b_3 &= 8(p_2 - 5) \\ b_2 &= 4(p_3 - 4p_1) \\ b_1 &= 2(p_4 - 3p_2 + 5) \\ b_0 &= p_5 - 2p_3 + 2p_1 \end{aligned} \tag{10}$$

and

$$\begin{aligned} c_5 &= 32 \\ c_4 &= 16q_1 \\ c_3 &= 8(q_2 - 5) \\ c_2 &= 4(q_3 - 4q_1) \\ c_1 &= 2(q_4 - 3q_2 + 5) \\ c_0 &= q_5 - 2q_3 + 2q_1 \end{aligned} \tag{11}$$

Here, p_i and q_i are coefficients of $p(z)$ and $q(z)$, respectively, where *i* refers to a polynomial term containing z^i. The p_0 and q_0 are always equal to one. For a given **a,** it is easy to show using Eq. (7a) that the remaining p_i and q_i can be obtained recursively through aloopofifrom 1 to 5 of

$$\begin{aligned} p_i &= a_i + a_{11-i} - p_{i-1} \\ q_i &= a_i + a_{11-i} - q_{i-1} \end{aligned} \tag{12}$$

The fast LSP conversion then uses the fact that each x associated with a zero of $p(z)$ or $q(z)$ causes *p(x)* or *q(x)* to be zero, respectively. Thus, the scheme applies values of x corresponding to ω in the LSP table (i.e., *LSPTable[j,i]*) to the polynomials $\tilde{p}(x)$ and $\tilde{q}(x)$, and observes for zero crossings. As before, j even and odd correspond to $\tilde{p}(x)$ and $\tilde{q}(x)$, respectively. For each j, the scheme then assigns certain *i* to LSP_j, such that x=*LSPTable[j,i]* is the closest x within the same j that causes a zero crossing of $\tilde{p}(x)$ or $\tilde{q}(x)$.

2.2.3. Ensuring LSP Stability.

We must have a scheme for robust representation of the LPC parameters, because they are very sensitive and the conversion to LSP parameters increases the sensitivity. Since $H(z)$ is a recursive filter, a distortion in **a** can easily move the poles of $H(z)$ to outside the unit circle of the z-plane, resulting in an unstable $H(z)$. The conversion to LSP further introduces more distortion due to quantization errors.

Fortunately, if the ordered values of ω_j are monotonically increasing (from 0 to π), the LSP method guarantees the stability of $H(z)$ [SoJu84]. Thus, before transmitting the LSP_j, the scheme verifies the ordered values of ω_j corresponding to LSP_j. If the ordered values violate the monotonicity, the scheme replaces it with a stable set of LSP_j form previous frame.

Sometimes, the pre-defined quantization steps can also create a stability problem. There are cases when some adjacent ω_j are too close together, so that for the given resolution, the table fails to distinguish them. Or, the ω_j may lie beyond the table coverage. In this situation, the fast LSP conversion usually gives incorrect, unstable LSP_j. An effort to avoid such cases is by expanding the bandwidth of **a** prior to LSP conversion process. Thus, instead of using **a,** the scheme use c, defined as

$$c_i = a_i \gamma^i \tag{13}$$

where γ is the expanding factor (typically set to 0.994), and *i* is an index from 1 to 10.

2.3. Codebook Parameter Searching

2.3.1. Searching Problem

To obtain the codebook parameters, the analysis searches for codebook 'parameters minimizing perceptual distortion

$$\|\mathbf{e}\|^2 = \|\mathbf{s} - \hat{\mathbf{s}}\|_W^2 = \left\|P_W[\mathbf{s} - \hat{\mathbf{s}}]\right\|^2 \tag{14}$$

where $\|\bullet\|$ denotes a norm (or magnitude) of a vector, and P_w represents a perceptual weighting filter defined as

$$P_W(z) = \frac{H(\frac{z}{\gamma})}{H(z)} \qquad 0 \le \gamma \le 1 \tag{15}$$

A typical γ is 0.8. (Such a $P_W(z)$ makes Eq. *(2) a* perceptual spectral-masking based measure rather than simply a pure Euclidean measure of waveform closeness). We call e the perceptual error vector.

The codebook parameters affect perceptual distortion in Eq. (14) through the excitation $\mathbf{t}$ and then $\hat{\mathbf{s}}$. A codebook consists of prototypes or ***codewords b,*** which are arrays of impulses ***b[n].*** Each codeword is indexed by a codebook entry called ***CBEntry.*** For each codebook, there is a gain table containing ***gain factors,*** which are real numbers. Each gain factor is indexed by a gain table entry called ***GainEntry. Thus*** for the ACB and SCB there ***are ACBEntry*** and ***SCBEntry,*** respectively, while for the ACB and SCB gain table entry there ***are ACBGainEntry*** and ***SCBGainEntry,*** respectively.

A set of those entries produces $\mathbf{t}$ according to

$$\mathbf{t} = \mathbf{b}_{(a)}(ACBEntry)g_{(a)}(ACBGainEntry) + \mathbf{b}_{(s)}(SCBEntry)g_{(s)}(SCBGainEntry) \quad (16)$$

The $\mathbf{b}_{(a)}(ACBEntry)$ and $\mathbf{b}_{(s)}(SCBEntry)$ ***are the*** ACB and SCB codewords pointed by ***ACBEntry*** and ***SCBEntry,*** respectively, while $g_{(a)}(ACBGainEntry)$ and $g_{(s)}(SCBGainEntry)$ are the ACB and SCB gain factors pointed by ***ACBGainEntry*** and ***SCBGainEntry,*** respectively. For a given s, the $\mathbf{t}$ produces $\hat{\mathbf{s}}$ and then e according to Eq. (2) and Eq. (14), respectively. Thus the search problem becomes: for a given ***searching target s,*** find ***ACBEntry, SCBEntry, ACBGainEntry,*** and ***SCBGainEntry*** corresponding to e that minimizes Eq. (16).

To solve the searching problem, there are several techniques such as those described in [KlKK90]. However, not all off them can be combined. We describe here fast searching algorithms that we actually use. Some are mandatory (implied by FS-1016), while some are our choice. We also discuss their consequences in the scheme.

2.3.2. Breaking the Frames into Subframes

One obvious way to reduce the computational cost for searching is by reducing the size of the codebooks, i.e., reducing the number of prototypes in the codebook. However, this approach increases the vector quantization error. To reduce the quantization error, one should reduce the length (i.e., dimension) of the prototype. However, this increases the bit requirement because we need more prototype to represent a segment of $\mathbf{t}$. FS-1016 solves this delicate balance by using a prototype length of **60** samples. This means, the searching target in one frame is split into four s in four ***subframes,*** and the scheme performs four searching processes to complete encoding of one frame, resulting in four sets of codebook entries. The SCB size can then be reduced to as low as **5** 12 while preserving natural speech quality.

It should be noted that since ACB is a codebook that actually represents a one adaptive tap, all pole pitch filter [Lang92], its size is not determined this way. The ACB size determines the range of pitch frequency it can cover. For an excitation ***x[n],*** the filter produces

$$y[n] = gy[n-d] + x[n] \quad (17)$$

with g as the filter coefficient (equivalent with ACB gain) and ***d*** is the tap position (equivalent with ACB entry). Varying ***d*** changes the pitch frequency (in Hz) according to

$$\text{Pitch Frequency} = \frac{\text{Sampling Frequency}}{d} \quad (18)$$

FS-1016 covers pitch frequency between 54 Hz to 400 Hz, requiring ***d*** to be between 20 to 147. Thus, we use an ACB size of 128. FS-1016 actually provides a size option of 256 to improve the pitch resolution in high frequency (associated with woman speakers). It is clear from Eq. (18) that the pitch resolution at higher frequency is coarser. The additional ACB entries are then added to improve the high frequency resolution. To reduce the computational cost, we did not use this option.

The subframe search approach also enable a smoother transition of LSP parameters through interpolation. Thus for each subframe i = 1, . . ., 4 , the scheme uses different $H(z)$ coming from interpolated LSP parameters defined as

$$\omega_j = \frac{9-2i}{8}\text{Previous}\omega_j + \frac{2i-1}{8}\text{Present}\omega_j \quad (19)$$

Thus the system must always keep the LSP parameters from the previous frame.

2.3.3. Combining Perceptual and Spectral Filters

We can reduce the computation cost by reducing the number of filters used during the search. To compute the perceptual distortion in Eq. (14), each prototype must pass through the LPC filter and the perceptual weighting filter. In the z-domain, the perceptual distortion vector is

$$\begin{aligned} E(z) &= P_w(z)\{S(z) - \hat{S}(z)\} \\ &= P_w(z)S(z) - P_w(z)H(z)T(z) \\ &= Y(z) - W(z)T(z) \\ &= Y(z) - X(z) \end{aligned} \quad (20)$$

where

$$Y(z) = P_w(z)S(z) \tag{21}$$

$$W(z) = P_w(z)H(z) = \frac{H(\frac{z}{\gamma})}{H(z)}H(z) = H_0\frac{z}{\gamma} \tag{22}$$

$$X(z) = W(z)T(z) \tag{23}$$

Observe that there is only one filtering $W(z)$ required now (i.e., Eq. (23)) for every prototype. As a new searching target, $Y(z)$ is calculated once only using Eq. (21), and then the search minimizes (in vectorial notations)

$$\|e\|^2 = \|y - x\|^2 \tag{24}$$

There is a slight problem of this approach if we calculate Eq. (23) in vector and matrix operations. In a matrix form, filter $W(z)$ is approximated by a 60 × 60 matrix W defined as

$$W = \begin{bmatrix} w[0] & 0 & \cdots & 0 \\ w[1] & w[0] & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ w[59] & w[58] & \cdots & w[0] \end{bmatrix} \tag{25}$$

where $w[i]$ are the impulse responses of $W(z)$, such that

$$x = wt \tag{26}$$

Unfortunately, the search results are good only if the CELP synthesizer also uses $H(z)$ in a matrix form, which is not the case. Let z be the zero response of $H(z)$ at the synthesizer, i.e., $z[n]$ are the output of the $H(z)$ when its input is zero for all subframe. In practice, z is not zero due to the non-zero contents of the $H(z)$ delay elements, resulting from the previous excitation. Thus, the actual output of the synthesizer is

$$\hat{s} = Ht + z \tag{27}$$

The analyzer must then introduce a compensation scheme such that we minimize Eq. (14) but still use combined filter W with Eq. (26). From Eq. (27) we have

$$P_w\hat{s} = P_wHt + P_wz = x + P_wz \tag{28}$$

Using the derivation in Eq. (20), we have

$$\begin{aligned} \|e\|^2 &= \|P_ws - x - P_wz\|^2 \\ &= \|P_w(s - z) - x\|^2 \\ &= \|\tilde{y} - x\|^2 \end{aligned} \tag{29}$$

Now, $\tilde{y}$ is the new searching target, defined as

$$\tilde{y} = P_w(s - z) \tag{30}$$

Let $e(CBEntry, GainEntry)$ be the perceptual error vector corresponding to a codebook entry $CBEntry$ and a gain entry $GainEntry$. Clearly minimizing

$$\|e(CBEntry, GainEntry)\|^2 = \|\tilde{y} - x(CBEntry, GainEntry)\|^2 \tag{31}$$

is equivalent to minimizing Eq. (20), with z has been taking into account. Figure 2 shows the new structure.

2.3.4. ***Serial Search***

To further reduce the computational cost, the scheme serially searches the ACB parameters before the SCB parameters. The system uses 5 12 and 128 entries for SCB and ACB, respectively, and 16 entries for each gain table. If the scheme has to search all codebooks simultaneously, it has to search through 512×128×16×16 = 16,777,216 entries. On the other hand, serial search works on 512×16 + 128×16 = 10,240 trials only.

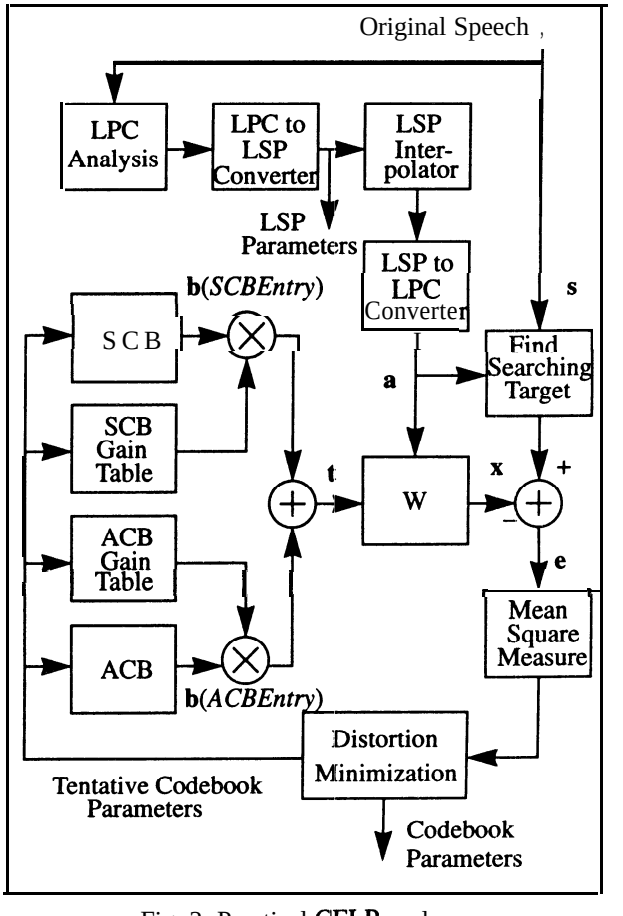


Fig. 2. Practical CELP analyzer.

Consequently, **ACB** and **SCB** searches differ in the searching targets. The searching target of ACB is $\tilde{\mathbf{y}}$ as defined in Eq. (30). The resulting ACB parameters alone can produce x according to Eq. **(26),** but they result in a high $\|\mathbf{e}\|^2$. The SCB parameters must then generate a signal that 'fills the gap' between $\tilde{\mathbf{y}}$ and such an x. Thus, $\tilde{\mathbf{y}} - \mathbf{Wt}$ becomes the SCB searching target, where **t** is obtained from **Eq.** (16) using newly obtain ACB parameters but without SCB parameters.

2.3.5. Joint Optimization Search

A ***joint optimization*** scheme suboptimally searches for **codebook** and gain entries in one process, thus further reducing the number of prototype trials. In minimizing $\|\mathbf{e}(CBEntry, GainEntry)\|^2$, the system should search through all combinations of ***CBEntry*** and ***GainEntry.*** However, the joint optimization scheme assigns an optimal ***GainEntry*** for each ***CBEntry, so*** that the scheme effectively searches for ***CBEntry*** only. In other words, instead of searching through 10,240 entries, the scheme only needs to search through **512+128** = 640 entries. This suboptimal solution saves computation in an order of magnitude. The basic approach is as follows.

1. For every **codebook** entry called ***CBEntry,*** compute v (sometime called the ***normalized x,*** i.e. the x obtained with unit gain, according to
$$v = W\,\mathbf{b}[CBEntry] \tag{32}$$
Here, the $\mathbf{b}[CBEntry]$ is a prototype in the **codebook** pointed by the ***CBEntry.*** This process is often called ***convolution.***
2. For ***every CBEntry,*** compute ***GainEntry*** associated with ***the CBEntry.*** Suppose g is the gain value which scales b to become **t.** Clearly,
$$\mathbf{x} = g\,\mathbf{v} \tag{33}$$
One way to minimize Eq. (31) is to maximize a Peak value defined in ***inner-product*** terms as
$$Peak = \langle \tilde{\mathbf{y}}, \mathbf{x} \rangle - \langle \mathbf{x}, \mathbf{x} \rangle = g\langle \tilde{\mathbf{y}}, \mathbf{v} \rangle - g^2 \langle \mathbf{v}, \mathbf{v} \rangle \tag{34}$$
To find the best g to maximize Eq. **(34),** we take a derivative of Eq. (34) with respect to g, and find its root. The root, which is the best g, turns out to be
$$g = \frac{\langle \tilde{\mathbf{y}}, \mathbf{v} \rangle}{\langle \mathbf{v}, \mathbf{v} \rangle} \tag{35}$$
Furthermore, ***GainEntry*** is now the index whose value in the gain table is the closest value to this g.
3. For ***every CBEntry,*** compute ***also the Peak*** value using Eq. (33).
4. Find the ***CBEntry*** that has the closest distance, that is one with the highest Peak value. This ***CBEntry*** and its associated ***GainEntry*** become the desired **codebook** parameters.

Notice that there are three main computational processes: the convolution to obtain v and the two inner products $\langle \tilde{\mathbf{y}}, \mathrm{v} \rangle$ and (v, v) . They are called many times, as many as the **codebook** size. Consequently, they are the bottleneck of the system.

2.3.6. Fast Convolution with Special Codebooks

The search scheme employs a fast convolution algorithm for the convolution in Eq. (32) by exploiting ***the overlapping*** property of the **codebook** elements **[KlKK90]. As** a result, some of the convolution results of an entry can be used to compute convolution of the next entry. Let us design an SCB such that all the prototypes' elements come from an array **r** having 1082 elements. Suppose the elements of a prototype pointed by ***CBEntry*** (i.e., $\mathbf{b}(CBEntry)$) ***are*** $b_{CBEntry}[i]$ with i = o, ... 59. Then we force the elements to **be**
$$b_{CBEntry}[i] = r[2(511 - CBEntry) + i] \tag{36}$$
It can be verified that the prototypes are overlapping, i.e., most elements of a prototype are also elements of another prototype in its neighborhood.

With this special SCB, we can obtain $v_{CBEntry}[i]$ using Eq. (32) as follows
$$v_{CBEntry}[i] = \sum_{j=0}^{59} w[j-i]\, b_{CBEntry}[j] \tag{37}$$
To simplify the notation, define $u(CBEntry,i,j)$ *as*
$$\begin{aligned} u\,(CBEntry,\ i,j) &= w[j-i]\, b_{CBEntry}[j] \\ &= w[j-i]\, r[2(511 - CBEntry) + j] \end{aligned} \tag{38}$$
We then have
$$\begin{aligned} v_{CBEntry}[i] &= \sum_{j=0}^{1} u\,(CBEntry,\ i,\ j) + \\ &\quad \sum_{j=2}^{61} u\,(CBEntry,\ i,\ j) - \\ &\quad \sum_{j=60}^{61} u\,(CBEntry,\ i,\ j) \\ &= \text{head term} + \text{middle term} - \text{tail term} \end{aligned} \tag{39}$$
It can be verified easily that the middle term is exactly $v_{CBEntry-1}[i]$ **because** of the overlapping property.

This remarkable fact leads to a fast iteration for convolution. Now, instead of performing 60 terms of multiply and accumulate (MAC) operations as implied by Eq. (37), the scheme calculates a $v_{CBEntry}[i]$ in 4 MAC only to obtain the head and tail terms, and uses the previously calculated $v_{CBEntry-1}[i]$ as the middle term. The computational cost reduction is by a factor of 15.

We can even avoid having to compute the tail term if we can afford having a long array **v'** and a short array **v"** of length 1082 and 60, respectively, as shown in the following modified joint-optimization algorithm.

1. We start with computing v_0 *[i]* using the old method (Eq. (37)) as a starting point for iteration. Store the results into an empty **v'** according to

$$v'[i] = v_0[i]; \qquad i = 0, \ldots. 59 \qquad (40)$$

2. Calculate *Peak* and *GainEntry* as in the joint optimization, and store them in *BestPeak* and *BestGainEntry*, respectively. Store also *CBEntry* (in this case is 0) into *BestCBEntry*.

   Then for *every CBEntry* = 1, ... 511, perform:

3. Calculate the 60 head terms and store it in **v"**.
4. Update the array **v'** according to

$$v'[i + 2CBEntry] \leftarrow v'[i + 2CBEntry] + v''[i] \quad (41)$$

5. Calculate *Peak* and *GainEntry* as in the joint optimization. However, get v from **v'** according to

$$v[i] = v'[i + 2CBEntry]; \qquad i = 0, \ldots. 59 \quad (42)$$

6. Compare *Peak* with a variable *BestPeak* (predefined as zero). If current *Peak* is larger than *BestPeak*, the scheme updates *BestPeak* with *Peak*, and stores *CBEntry* and *GainEntry* in *BestCBEntry* and *BestGainEntry*, respectively.

After performing those steps for all entries, the desired parameters are available in *BestCBEntry* and *BestGainEntry*.

Further cost reduction is due to the fact that FS-1016 SCB uses $b_{CBEntry}$ [i] that is not only overlapping but also sparse (77% of the elements are 0) and ternary (i.e., the elements takes values -1, 0, and 1 only). Thus before calculating the head terms in the Step 3 above, the scheme checks if $b_{CBEntry}[j]$ is zero. In such cases, 60 computations of the term using this $b_{CBEntry}[j]$ in Step 3 are skipped. The scheme should have 77% of such cases.

With ternary $b_{CBEntry}[j]$, multiplications in computing the head terms are not necessary anymore because multiplication by 1 and -1 are equivalent with changing sign only.

Although the above example is derived for the SCB search, the ACB search can also use fast convolution. Since ACB is actually a one-tap, all-pole filter, the overlapping property is inherent in the ACB. However, the ACB elements are not ternary nor sparse, thus both calculation of the head terms and multiplications cannot be omitted. But, the calculation is fast already, because the number of MAC in its head term is one only (except in some special cases at the lower entries), instead of two as in the SCB. Furthermore, the size of the ACB we use is 128 as opposed to 5 12 of the SCB.

It should be clear that this fast convolution works only if we use $W(z)$ in a matrix form, otherwise we cannot have Eq. (37) and the rest of its derivations.

2.3.7. Delta Coding for ACB Parameters

Further computation reduction is possible for ACB search. Here we utilize the fact that human pitch does not suddenly change within two subframes (15 ms). This means we expect that the difference between selected codebook entries of consecutive subframes can be less than 64 entries. Thus *we* can employ *delta coding* that codes the entry difference only. Such coding needs a reference point. The FS- 1016 uses ACB entries of odd subframes as the references and delta codes the even subframes, i.e., the entry of the second or the fourth subframe is represented by the difference between the actual entry and the previous-subframe entry. This scheme reduces the computation because the even search routine operates on a subset of the ACB only (64 entries instead of 128 entries). This scheme also reduces the bit rate since the number of bits to represent the difference is less than that to represent the actual entry.

2.4. Speech Synthesis

2.4.1. Synthesis Process

A CELP synthesizer reconstructs 240 samples of s from a set CELP parameters. In principle, the synthesizer must first construct the filter $H(z)$ using the interpolated LSP parameters. The synthesizer then computes the excitation impulses **t** for one subframe using the codebook parameters according Eq. (17). Finally, it applies the excitation impulses **t** to the filter $H(z)$ to synthesize 60-element speech $\hat{\mathbf{s}}$ using Eq. (2). Repeating the process three more times results in a complete 240 elements of $\hat{\mathbf{s}}$.

Since most of the **steps have** been explained, we just describe here the *conversion* of LSP to LPC parameters.

2.4.2. LSP to LPC Conversion

We want **to reconstruct a from** the interpolated **LSPs** ω_j. Let us define xp and xq according to

$$\left.\begin{aligned} xp_i &= \omega_{2i} \\ xq_i &= \omega_{2i+1} \end{aligned}\right\} \quad i = 0, \ldots 4 \tag{43}$$

The following steps then convert the LSP:

1. Recover the array b as in Eq. (10) according to the following equations

$$\begin{aligned} b_5 &= 32 \\ b_4 &= -b_5 \sum_{i=O}^{4} xp_{i+1} \\ b_3 &= b_5 \sum_{i=1}^{4} \sum_{j=i+1}^{5} xp_i xp_j \\ b_2 &= -b_5 \sum_{i=0}^{3} \sum_{j=i+1}^{4} \sum_{n=j+1}^{5} xp_i xp_j xp_m \\ b_1 &= b_5 \sum_{i=1}^{2} \sum_{j=i+1}^{3} \sum_{m=j+1}^{4} \sum_{n=m+1}^{5} xp_i xp_j xp_m xp_n \\ b_0 &= -b_5 (xp_1 xp_2 xp_3 xp_4 xp_5) \end{aligned} \tag{44}$$

2. Recover the coefficients of $p(z)$ according the following equations

$$\begin{aligned} p_1 &= \frac{b_4}{16} \\ p_2 &= \frac{b_3 + 40}{8} \\ p_3 &= \frac{b_2 + 16p_1}{4} \\ p_4 &= \frac{b_1 + 6p_2 - 10}{2} \\ p_5 &= b_0 + 2p_3 - 2p_1 \end{aligned} \tag{45}$$

3. Recover array c as in Eq. (11) according the following equations

$$\begin{aligned} c_5 &= 32 \\ c_4 &= -c_5 \sum_{i=O}^{4} xq_{i+1} \\ c_3 &= c_5 \sum_{i=1}^{4} \sum_{j=i+1}^{5} xq_i xq_j \\ c_2 &= -c_5 \sum_{i=0}^{3} \sum_{j=i+1}^{4} \sum_{n=j+1}^{5} xq_i xq_j xq_m \\ c_1 &= c_5 \sum_{i=1}^{2} \sum_{j=i+1}^{3} \sum_{m=j+1}^{4} \sum_{n=m+1}^{5} xq_i xq_j xq_m xq_n \\ c_0 &= -c_5 (xq_1 xq_2 xq_3 xq_4 xq_5) \end{aligned} \tag{46}$$

4. Obtain set of *q(z)* coefficients, according the following equations

$$\begin{aligned} q_1 &= \frac{c_4}{16} \\ q_2 &= \frac{c_3 + 40}{8} \\ q_3 &= \frac{c_2 + 16q_1}{4} \\ q_4 &= \frac{c_1 + 6q_2 - 10}{2} \\ q_5 &= c_0 + 2q_3 - 2q_1 \end{aligned} \tag{47}$$

5. Finally, use p and q to construct a by inverting Eq. **(12),** as follows

$$\begin{aligned} a_0 &= 1 \\ p_0 &= q_0 = 1 \\ a_i &= \frac{p_{i-1} + p_i + q_i - q_{i-1}}{2}; \quad i = 1, \ldots . 5 \\ a_{11-i} &= \frac{p_{i-1} + p_i + q_i - q_i}{2}; \quad i = 1, \ldots . 5 \end{aligned} \tag{48}$$

3. COMPUTER IMPLEMENTATION

We can now translate the above algorithm to a computer implementation. We have coded the procedures in ANSI C routines. We briefly describe the actual program to show how a CELP system actually uses the procedures. Details of routines for **codebook** searching are presented in **[GrLK93].**

3.1. LSP Analysis

First, a routine PCMtoFloat converts the speech samples s into a floating point form, since s usually comes from an analog-to-digital converter with integer data format, while floating-point computation is preferred to reduce the distortion caused by finite-length registers. A routine AnalyzeLPC then extracts **a** from s, explained in Section 2.2.1. Prior to converting **a** to LSPs, the scheme calls a routine ExpandBandwidth to expand the bandwidth of a_i using Eq. (13) with an expanding factor γ of 0.994. This procedure ensures that **a are** within the range of the LSP table. The scheme calls ConvertLPCToLSP routine to obtain LSP_j from **a,** according to Section 2.2.2. Finally, a routine CheckLSPStability verifies the monotonicity of the LSP_j before allowing them to be used (see section *2.2.3).*

3.2. Codebook Searching

A computer routine called CodebookSearching finds the ACB and SCB parameters. First, we must construct a 60×60 matrix *W* representing $W(z)$ *(see Eq.* (25)). The scheme starts with obtaining **a.** An Interpo lat eLSP routine provides LSPs for each individual subframe by interpolation using Eq. (19). The filter *W* practically requires **a** instead of LSPs, thus the scheme calls ConvertLSPtoLPC routine for the conversion (see Section 2.4.2). An ExpandBandwidth routine then performs Eq. (13) to generate c, with an expanding factor y of 0.8. It is easy to show using Eq. (22) that $W(z)$ is equivalent with $H(z)$ with c replaces **a.** Furthermore, to represent the filter *W(z), W* must contain the impulse responses, w_i, of the filter, as shown in Eq. (22). A FindImpulseResponse routine provides such elements.

Having constructed *W,* the scheme prepares for ACB parameter searching. The FindACBSearching-Target routine determines the ACB searching target $\tilde{\mathbf{y}}$ for the current subframe using Eq. (30).

If the subframe is odd, i.e., the first or third subframe, the scheme calls the ACBSearchingOdd routine to get the ACB parameters, otherwise the ACBSearchingEven performs that function. The scheme then computes the searching target of SCB searching, by calling FindSCBSearchingTarget. The SCB-Searching obtains the SCB parameters and stores them in an output buffer. The routine now has a complete set of the codebook parameters.

Before the loop proceeds for the next subframe, it must prepare and update the system states. First, an Updat eACB routine updates the contents of the ACB with new values from the excitation impulses to imitate the effects of delay elements in an all-pole, one-tap pitch filter. Second, the delay elements of $H(z)$ must also be updated according to those of the CELP synthesizer. At this phase, the synthesizer has stored values in its delay elements which has an additive effect to the synthetic speech produced later in the next subframe. The Get - De layElement s tracks those values, which are later used by the next-subframe FindACBSearching-Target to compensate the additive effect represented by the zero response, as discussed in Section 2.3.3. Finally, if the subframe is even, i.e., the second or fourth subframe, the Del t aEncoding ACB routine encodes the ACB entry using a delta coder.

Having obtained the LSPs and all entries for ACB and SCB for one frame, the scheme collects them in an FS-1016 data stream for transmission, by calling ConvertToDataStream. A routine UpdatePreviousLSP updates the contents of previous LSPs with the newly obtained LSPs to be used for interpolation (using Eq. (19)) and stability checking in the next frame.

3.3. Speech Synthesis

A synthesis program converts each FS- 1016 data stream into a frame of speech. First, a routine ConvertFromStream unpacks; the LSPs and the entries of ACB and SCB from the data stream. Since two of the ACB entries are delta coded, a routine DeltaDecoding obtains the actual entries.

As in the case of codebook searching, the synthesis performs a loop for four consecutive subframes. The loop starts with InterpolateLSP to obtain smooth transition of the LPC filter *H(z),* using Eq. (19). A routine ConvertLSPtoLSP provides **a** from the LSPs to construct $H(z)$ (see Section 2.4.2). To get the excitation impulses **t,** the loop calls Updat eACB, which computes **t** using the ACB and SCB entries, and also updates ACB using the resulting **t.** Finally, a routine GetDelayE lement s applies the **t** to $H(z)$ to produce the speech, and at the same time, updates the delay elements of $H(z)$ to be used later for the next *H(z).*

Before the process continues to the next frame, a routine UpdatePreviousLSP updates the contents of previous LSPs with the current LSPs.

4. PERFORMANCE

The algorithm presented here is fast enough for practical uses, such as store-and-forward communication, voice-mail, and multimedia. We have ported the computer program for various platforms, including TMS C30, IBM PC, SUN workstations, and IBM PowerPC based workstation. It has also been ported as a dynamic link library (DLL) for Windows 3.1, ready to be used for various speech applications.

Figure 3 shows a simple CELP compression application as an example of accessing the DLL.

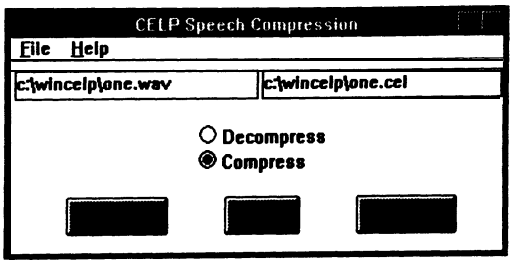


Fig. 3. A simple Windows 3.1 CELP system utilizing the CELP dynamic link library.

Table 1 shows that the execution time is within a reasonable range. On the IBM Power PC workstation, the algorithm run faster than real-time (0.85 real-time for both analysis and synthesis). The execution time of the **C30** implementation is approximately two to three

the routines require between five times to twice real-time requirement. **Mhz** IBM-PC 486DX requires approximately 14 times real-time.

up the **codebook** searching, the synthesized speech still has high intelligibility and natural quality **[Lang92].** The results from a Fairbanks rhyme test show an intelligibility score of more than 95% word correct identification. Furthermore, subjective and objective tests using male spoken Harvard sentences result in a mean opinion score (MOS) of 3.21 and a segmental signal-to-noise ratio (SEGSNR) of 10.10 **dB,** respectively.

platforms, in terms of **%** real-time.

	Analysis Time (**%** real-time)	Synthesis (**%**
PC **486DX/33**	1304	**23**
SUN **Sparc** 2	**445**	12
SUN **Sparc** 5	221	5
PC-AT/ TMS **C30**	**220**	**5**
PowerPC (Power 590)	**83**	**2**

Furthermore, the size of the executable file is small. The **C30** program and data require less than 11 Kwords of memory. The size of the SUN version executable file is 64 Kbytes. The algorithm can be coded modularly in C to enable tailoring it to another application.

However, the fast algorithms is quite complex, i.e., it involves many processes, loops, and **variables. The** efforts in reducing the computation time results in increasing the memory requirement to hold look-up tables and codebooks. The algorithm also reduces the overhead in data transfers by fixing the locations of arrays and globally using them. This increases the complexity, because data may be altered by several different processes, which means there are many processes that should be considered simultaneously.

In most platforms, a real-time application still requires faster processors. Our observation on the **C30** program reveals that the **codebook** searching consumes 218% of real-time requirement, i.e., 2.18 seconds of **codebook** searching are required for every second of speech. As shown in Table 2, this results from the inner products inside the joint optimization scheme, which consumes **111%** of the real-time requirement. This part should become the main attention to improve the execution **speed.**

However, it should be noted that the synthesis part requires only 2 to 23% of the real-time requirement of execution time in all platforms. This means the system can easily perform real-time playback. This asymmetric type of systems (i.e. systems with easy playback) has found a wide range of applications, such as in broadcasting, database, library, and CD-ROM based multimedia.

Table 2. Computation time requirements of some most demanding routines in **TMSC30.**

Processes	**%** Real Time
Inner Products	**111**
Joint Optimization	148
Codebook Search	218
SCB Searching Target	**6**

5. DISCUSSION

This paper has described an efficient algorithm and its implementation of the CELP speech processing system. Near real-time implementation is possible using fast extraction of LSP parameters, fast searches of ACB and SCB parameters, and CELP synthesis. The **codebook** searches employ the joint optimization scheme, which consumes the largest block of the **codebook** searching computation due to a combination

of the complexity of this routine and the large number of times it is called by the ACB and SCB searching algorithms. The algorithm allows high quality speech to be achieved with a bit rate of as low as 4.8 kHz. The algorithm can be readily used for CELP implementations, such as on (i) high quality low-bit rate speech transmission in point-to-point or store-and-forward (network based) mode, and (ii) efficient speech storage in speech recording or multimedia databases.

We are currently seeking hardware implementation to reduce not only the execution time, but also the physical size of the actual implementation, We are studying the algorithm for the purpose of casting some of its parts to silicon. At this stage, a full hardware implementation is premature since the optimality is not clear. However, we should focus on casting the inner products and convolution processes that have become the algorithm bottleneck. Implementing the inner product process in dedicated hardware is attractive, because it has a simple computational structure, i.e., a regular multiply and accumulate process of 60 terms. The convolution of SCB elements in Eq. (39) is also attractive for hardware implementation because the SCB elements are predefined. As explained in Section 2.3.6, the ternary property simplifies the convolution into an addition/ subtraction process with a branch controlled by SCB elements, making it easier to implement in hardware.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, the Manitoba Telephone System (MTS), and now by the Telecommunication Research Laboratories (TRLabs). One of the authors (AL) wishes to thank IUC-Microelectronics ITB, Laboratory of Signals and Systems ITB, and PT INTI Persero, Bandung, Indonesia, for their support in this research work.

REFERENCES

[Atal86] B. S. Atal, "High quality speech at low bit-rates: Multi-pulse and stochastically excited linear predictive coders", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* (Tokyo, Japan), IEEE CH2243-4/86, pp. 168 1- 1684, 1986.

[CaTW90] J. P. Campbell, Jr., T. E. Tremain, and V. C. Welch, "The proposed Federal Standard 1016 4800 bps voice coder: CELP", *Speech Technology,* pp. 58-64, Apr./May 1990.

[GrLK93] W. Grieder, A. Langi, and W. Kinsner, "Codebook searching for 4.8 kbps CELP speech coder", in *Proc. IEEE Wescanex 93,* pp. 397-406.

[Hahe90] R. Ragen, and P. Hedelin, "Low bit-rate spectral coding in CELP, a new LSP method", in *Proc IEEE Int. Conf. Acoust., Speech, Signal Processing,* IEEE CH2847-2/90, pp. 189- 192, 1990.

[JaJS93] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception", *Proceeding of IEEE,* vol. 8 1, no. 10, pp. 1385-1422, October 1993.

[KaRa86] P. Kabal, and R. P. Ramachandran, "The computation of line spectral frequencies using Chebyshev polynomials", *IEEE Trans. ASSP,* vol. ASSP-34, no. 6, pp. 1419-1426, December 1986.

[KlKK90] W. B. Kleijn, D. J. Krasinski, and R. H. Ketchum, "Fast Methods for the CELP speech coding algorithm", *IEEE Trans. ASSP,* vol 38, no. 8., pp. 1330- 1342, August 1990.

[LaKi90] A. Langi and W. Kinsner, "CELP High-quality speech processing for packet radio transmission and networking", *ARRL 9th Computer Networking Conf.,* pp. 164-169, 1991.

[LaKi91] A. Langi and W. Kinsner, "Design and Implementation of CELP Speech Processing System using TMS320C30", *ARRL 10th Computer Networking Conf.,* pp. 87-93, 1991.

[Lang92] A . Langi, "Code-Excited Linear Predictive Coding for High-Quality a.nd Low Bit-Rate Speech", *M.Sc. Thesis,* The University of Manitoba, Winnipeg, MB, Canada, 138 pp., 1992.

[Pars86] T. W. Parsons, *Voice and Speech Processing.* New York: McGraw-Hill, 402 pp., 1986.

[Proa83] J. G. Proakis, *Digital Communication. New* York: McGraw-Hill, 608 pp., 1983.

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Veterling, and B.P. Flannery. *Numerical Recipes in C.* (2nd ed) New York, NY Cambridge University Press, 1992.

[ScAt85]M. R. Schroeder, and! B. S. Atal, "Code excited linear prediction (CELP): high quality speech at very low bit rate," in *Proc. IEEE Znt. Conf. Acoust., Speech, Signal Processing,* IEEE CH2 118-8/85, vol 1, pp. 937-940, 1985.

[SoJu84] F. K. Soong, and B. -H. Juang, "Line Spectrum Pair (LSP) and speech data compression", in *Proc IEEE Int. Conf. Acoust., Speech, Signal Processing,* IEEE CH1945-5/84, pp. 1.10.1-1.10.4, 1984.

WAVELET COMPRESSION FOR IMAGE TRANSMISSION THROUGH BANDLIMITED CHANNELS

A. Langi, VE4ARM, and W. Kinsner, VE4WK

Department of Electrical and Computer Engineering,
and Telecommunications Research Laboratories
University of Manitoba
Winnipeg, Manitoba, Canada **R3T 5V6**
Tel.: (204) 474-6992; Fax: (204) 2750261
eMail: kinsner@ee.umanitoba.ca

ABSTRACT

This paper studies an image compression scheme using **wavelets** for image transmission through **bandlimited** channels. During encoding, the scheme first transforms the image to a **wavelet** domain and then compresses the **wavelet** representation into an image code. Conversely, during decoding, the scheme first decompresses the image code into **wavelet** representation and then transforms it back to the original domain. The **wavelet** transform is explained from a perspective of signal representation in $L^2(R)$ space. The transform is further computed using a pyramidal algorithm. The compression is possible because (i) the **wavelet** representation has many small values that can be coded using fewer bits, and (ii) the **wavelet** basis functions are localized in space and frequency domains such that an error in the **wavelet** representation only locally affects the image in those domains. An experiment has been performed to compress two **256×256** greyscale images on such a scheme through (i) a transformation using a simple **wavelet** called DAUB4, (ii) redundancy removal by truncation the small-valued **wavelet** representation, and (iii) a ZIP compression (based on Shannon-Fano and Lempel-Ziv-Welch techniques). The results show that highly truncated **wavelet** representation (2 90%) still provides good image quality (PSNR ≥ 30dB) at less than 2 bits per pixel (bpp). Severe truncation still preserves general features of the image.

1. INTRODUCTION

Image compression is a problem of considerable importance because it leads to efficient usage of channel bandwidth and storage during image transmission and storage, respectively. There are many applications that require image transmission, such as high-definition television (HDTV), videophone, video conferencing, interactive slide show, facsimile, and multimedia **[AnRA91], [Kins91], [Prag92], [JaJS93].** However, images require many data bits, making it impossible for real-time transmission through bandlimited channels, such as 48 **kbit/s** and 112 **kbit/s** integrated services digital network (ISDN), as well as 9.6 **kbit/s** voice-grade telephone or radio channels. Non real-time transmission on such channels also takes a long time. For example, a **256×256** pixel greyscale still image with 8 bits per pixel (bpp) requires 65,536 bytes. Transmission of such an image over the voice-grade line would require at least 54.61 s. Furthermore, one HDTV format needs **60** frames of 1280x720 pixels per second. Using 24 bpp color pixels, this HDTV format would require a channel capacity of 1,440 Mbits/s.

Image compression can improve transmission performance by reducing the number of bits that must be transmitted through the channel. As shown in Fig. 1, an image compressor compactly represents the image prior to channel encoding. At the receiving end, the demodulation and signal decoding obtain the compressed image, and an image decompressor converts the compressed image back to a **viewable** image. The transmission now requires a shorter time because the number of bits that must be transmitted has been reduced. Image compression also reduces the storage space requirement for image storage.

Although there exists several image compression methods, we still need better ones because current methods are still inadequate, especially for image **transmis-**

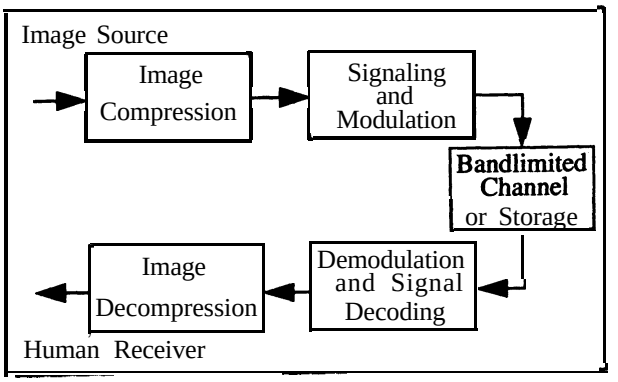


Fig. 1. Efficient image transmission system.

sion through bandlimited channels. For example, a sophisticated joint photograph expert group (JPEG) method needs 1.6 bpp for 24 bpp color images. This means that HDTV with 30 frames per second would still require a 4.423 Mbit/s channel capacity. Low bit rate compression methods have been relying on *lossy* type of compressing data, in which reducing the bit rate has caused a removal of some image information. Thus, there is a limit on how far a method can reduce the bit rate, while keeping information distortion sufficiently low.

The performance measurement of a compression technique is based on how successful it is to reduce the bit rate while maintaining image quality [Kins91]. The other performance aspects are algorithm complexity and communication delay [JaJS93]. This paper mainly focuses on the interplay between the first two aspects: reducing the bit-rate and maintaining quality.

The wavelet technique (or subband coding in general) has been emerging as an important class of image compression, in that it has an efficient representation with few visible distortions [Kins91], [Mall89], [JaJS93]. It is more promising than the JPEG standard, due to (i) a match between wavelet processing and the underlying structure of visual perception model, and (ii) a match between characters of wavelet analysis and natural images, i.e., higher frequency tends to have shorter spatial support [JaJS93]. They are also attractive because wavelet techniques are quite fast and easy to implement. Furthermore, wavelet techniques can provide special representation such as multi-resolution encoding [Mall89].

This paper shows the potential of using wavelet representation for image compression. The wavelet representation is introduced through a signal representation theory, and computed though a wavelet transform pair in Section 2. Section 3 shows a wavelet compression scheme based on the wavelet transform pair to convert spatial-domain images to and from the wavelet domain. The scheme then compresses the image in wavelet domain using a combination of lossless (no information loss) or lossy (some minor losses) techniques of data compression. Preliminary experiments on two 256x256 greyscale images (`lena. img` and `camera. img`) using a simple wavelet (DAUB4) and a simple compression (truncation and ZIP) show the potential of the scheme. Those images are used because they are standard and easily obtained by other researchers. At 4 bpp, the distortion is almost unnoticeable. At 2 bpp, the distortion is noticeable, but the errors are of salt and pepper noise type such that a simple filtering may be able to eliminate them. At highly compressed images, the details are lost but the general features are still preserved.

2. WAVELET TRANSFORM OF SIGNALS

The purpose of this section is to construct a discrete *wavelet transform* (DWT) for discrete signal representation. This construction is explained through a theory of signal representation in a special space, called $L^2(R)$, which is a space for all signals with bounded energy (square integrable) [Mall89]. This restriction does not really affect the application generality because most of the signals that are of interest to us are in that space. Although we use a one-dimensional signal as an example, we can extend the results to two-dimensional signals such as images.

2.1. Signal Representation in $L^2(R)$

In *L*(R),* a signal $x(t)$ can be represented in various domains using various sets of basis functions. Let a domain be spanned by basis functions $\varphi_i(t)$, having reciprocal signals $\theta_i(t)$, where $i \in Z$ (integer numbers). The representation of $x(t)$ in this domain is a set of scalars α_i [Fran69], satisfying

$$\alpha_i = \langle \mathbf{x}, \theta_i \rangle \tag{1}$$

where $\langle \bullet, \bullet \rangle$ is an inner product, defined in *L*(R)* as

$$\langle \mathbf{x}, \theta_i \rangle = \int_R x(t)\, \theta_i(t) dt \tag{2}$$

between signals $x(t)$ and $\theta_i(t)$. The $x(t)$ can be reconstructed back through

$$x(t) = \sum_i \alpha_i \varphi_i(t) \tag{3}$$

Thus $x(t)$ is a linear combination of $\varphi_i(t)$

It may happen that the basis are not countable. In other word, we have $\varphi(\tau,t)$ and $\theta(\tau,t)$, called *basis kernel,* instead of $\varphi_i(t)$ and $\theta_i(t)$. In such a continuous case, the new representation of $x(t)$, which is called $u(\tau)$, becomes

$$u(\tau) = \int_R x(t)\, \theta(\tau, t) dt \qquad t, \tau \in R \tag{4}$$

and the reconstruction becomes

$$X(t) = \int_R u(t)\, \varphi(t, \tau)\, d\tau \tag{5}$$

In many applications, it is desirable to have the same set for basis and the reciprocal, or equivalently $\varphi_i(t)$ is

equal to $\theta_i(t)$, because of the difficulties obtaining and dealing with certain $\theta_i(t)$. In this case, the basis functions are ***self-reciprocal,*** or ***orthogonal.*** Let ***the nom*** of a **signal** $f(t)$ be defined as

$$\|f(t)\| = \sqrt{\langle f(t), f(t) \rangle} \tag{6}$$

If the norm of each basis function is one, the basis **functions** are ***orthonormal.***

There are many basis functions available. In fact, there is a very large number of them. However, only a few of them are useful, including wavelets.

2.2. Wavelets as Basis Functions

As basis functions, **wavelets** have several special properties. First, in the frequency domain, a **wavelet** can be seen as a special **bandpass** signal. A **wavelet** operating at a higher frequency has a wider bandwidth. The bandwidth is proportional to the centre frequency of the signal. This is useful for short-time signal analysis because **wavelets** can provide enough analysis resolution in both original and frequency domains. Second, the **wavelets** in a set of basis functions are scaled (by a factor ***a)*** and translated (by a time-shift parameter ***b)*** versions of a single **wavelet** prototype $\psi(t)$ **[RiVe91].** In a mathematical notation, each **wavelet** is in the form of

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{7}$$

where a is the scaling factor and b is the translation parameter. Thus, the **wavelets** have self similarity, and useful in revealing self-similarity aspect of signals. This also leads to fast algorithms. Third, in contrast with Fourier representation, there are more than one **wavelet** prototypes. In Fourier representation, the prototype is e^{jwt} only. Thus the same **wavelet** tools can be applied to many different sets of **wavelet** basis.

The selection of a and b leads to a different class of **wavelet** representation. In general, the $\psi_{a,b}$ are not orthogonal because a and b can be any real, continuous value. In this case, applying Eq. (2) through (6) is more difficult because we must find the reciprocal basis for each **wavelet** set. However with some restrictions, we can use those equations as if the **wavelets** were orthogonal basis functions. The restrictions are that the **wavelet** prototype $\psi(t)$ must be (i) finite energy and (ii) **bandpass** (no DC component) **[RiVe91].** In this situation, if we use **Eq.** (5) and **(6), *we*** have a ***continuous wavelet trans-*** form (CWT) that operates on continuous signal using **wavelet** basis kernel. Notice that some modification is needed, because we have two parameters a and b instead of just one τ as in the Eq. (5) and Eq. (6). We then have

$$u(a,b) = \int_R x(t)\,\psi_{a,b}(t)\,dt \qquad t, a, b \in R \tag{8}$$

and

$$X(t) = \int_R \left(\int_R u(a,b)\,\psi_{a,b}(t)\,db \right) da \qquad t, a, b \in R \tag{9}$$

It is important to notice that by changing the parameter a and b, ***we*** can position a **wavelet** in any location in the time-frequency plane (phase plane). The parameter a is a scaling parameter which changes the frequency positions of the **wavelet.** A larger a results in a lower center frequency, smaller bandwidth, and larger time support of the **wavelet** However, in a logarithmic **frequency** scale, different values of a result in different **bandpass** signals of the same bandwidth. Furthermore, parameter b is a time shift parameter. It changes the time position of the **wavelet.** Thus **wavelet** analysis reveals both time and frequency characteristics of the signal.

2.3. Wavelet Series

We **may** want to use discrete a and b because **continuous** a and b lead to redundant representations. One optimal selection is a dyadic sequence of $a = 2^j$ and $b = 2^j k$ **[Mall89],** resulting in orthogonal **wavelets**

$$\psi_{j,k}(t) = \sqrt{2^{-j}}\,\psi(2^{-j}t - k) \tag{10}$$

We can now operate using Eq. (2) and (4). As before, **Eq.** (2) and (4) must be modified to use these basis signals because they have two integer indices, j and k, instead of one i. This modification results in a **wavelet** series representation.

2.4. Discrete Wavelet Transform

It is natural to extend the **wavelet** series for representing discrete signals using countable, discrete basis, that leads to DWT. Here, a certain selection of **wavelet** prototype and time-scale parameters leads to orthonormal **wavelets. Mallat** uses multi-resolution signal decomposition **[Mall89]** to obtain DWT, outlined here. The previous equations are still useful with slight modifications. Since we deal with countable basis functions, we use **Eq. (2) and (4)** as our basic transform pair, with Eq. (10) as the source of the basis. Furthermore, for digital signals, $x(t)$ is represented by $x[n]$, where $n \in Z$ (integer numbers).

The DWT can be developed through a multiresolution signal decomposition. Let us introduce a space V_0 which is a **subspace** of $L^2(R)$. ***The*** signal $x(t)$ lies in this space. The signal can be decomposed into several **sig-**

nals, which later called **wavelet** decomposition of $x(t)$ (analogy to Fourier decomposition of a signal into its frequency components). Conversely, we can use the decomposed signals to reconstruct $x(t)$ without any loss. The decomposed signals exist in decomposed subspaces created in a ***pyramidal*** structure. To explain this space structure, we first decompose V_0 into two orthogonal complement spaces called V_1 and O_1 denoted as

$$V_1 \oplus O_1 = V_0 \tag{11}$$

Orthogonal complement means

$$\begin{aligned} V_1 \cup O_1 &= V_0 \\ V_1 \cap O_1 &= \{\mathbf{0}\} \\ \text{VE } V_1, \mathbf{o} \in O_1 &\Rightarrow \langle \mathbf{v}, \mathbf{o} \rangle = 0 \end{aligned} \tag{12}$$

where $\{\mathbf{0}\}$ is a null set. Second similar decomposition is from V_1 into V_2 and O_2. The decomposition continues down to Jth decomposition, where V_{J-1} is decomposed into V_J and O_J, where $J \in Z, J > 0$. ***Thus,*** for every j, where $j \in Z$, $J \le j \le 1$, and $J > 0$, we have

$$V_j \oplus O_J = V_{j-1} \tag{13}$$

Mallat shows that there exists $\psi(t)$ such that ψ_j, k(f) as defined in Eq. (10) are basis functions of O_j. The basis functions are orthonormal in both O_j and $L^2(R)$ [Mall89]. Furthermore, **Mallat** showed that there exists ϕ (t) in the space $L^2(R)$ such that

$$\phi_{j,k}(t) = \sqrt{2^{-j}}\phi(2^{-j}t - k) \tag{14}$$

are orthonormal basis of V_j.

We are now ready to define the DWT. Let $x[n]$ be a digital signal of limited energy, i.e.,

$$\sum_n |x[n]|^2 < \infty \tag{15}$$

(This implies x [n] $\in l^2(Z)$). Let also continuous orthonormal signals $\{\psi_{j,k}(t)\}$ and $\{\phi_{j,k}(t)\}$ ($\in R$ and j, $k \in Z$) span $L^2(R)$. Finally, associate $x[n]$ with f(t) $\in L^2$ (R) according to

$$f(t) = \sum_k x[k]\,\phi_{0,k}(t) \tag{16}$$

The **DWT** of $x[n]$ is then a mapping from $l^2(Z)$ to $l^2(Z^2)$ **resulting** in a set of real numbers ($c_{j,k}$,$d_{J,k}$) (called **wavelet** coefficients), according to

$$c_{j,k} = \langle f(t), \psi_{j,k}(t) \rangle \equiv \int_{-\infty}^{\infty} f(t)\,\psi_{j,k}(t)dt \tag{17a}$$

$$d_{J,k} = \langle f(t), \phi_{J,k}(t) \rangle \equiv \int_{-\infty}^{\infty} f(t)\,\phi_{J,k}(t)dt \tag{17b}$$

as in Eq. (1).

In practice, we restrict $x[n]$ to be of finite length, with N elements. In this case, J is any integer between 1 and $\log_2 N$. (In this work, we set J to $(\log_2 N)-1$, thus the description of DWT in [Pres91a] is directly applicable). Indexj is called ***scale,*** ranging from 1, 2, ..., to J, while k is 0, 1, ... to $(2^{-j}N)-1$. Although DWT can be defined for complex signals, we have limited the Eq. (2) and Eq. (3) to real input and basis signals only.

Orthonormality **of** signals $\{\psi_{j,k}(t)\}$ and $\{\phi_{j,k}(t)\}$ implies that the inverse DWT gives back $x[n]$ from $\{c_{j,k}\, d_{J,k}\}$ **through**

$$f(t) = \sum_{j=1}^{J} \sum_{k=0}^{2^{-j}N-1} c_{j,k}\psi_{j,k}(t) + \sum_{k=O}^{2^{-J}N-1} d_{J,k}\phi_{J,k}(t); \tag{18}$$

as in **Eq. (3),** and then

$$x[n] = \langle f(t), \phi_{0,n}(t) \rangle \tag{19}$$

2.5. Fast Pyramidal Algorithm for DWT

Given a discrete signal $x[n]$, one can actually compute the basis inner products in Eq. (17) using a matrix **multiplication,** according to

$$\begin{bmatrix} c_{0,0} \\ \cdots \\ c_{j,k} \\ \cdots \\ d_{J,k} \end{bmatrix} = \begin{bmatrix} \psi_{0,0}[0] & \cdots & \psi_{0,0}[N-1] \\ \cdots & & \cdots \\ \psi_{j,k}[0] & \cdots & \psi_{j,k}[N-1] \\ \cdots & & \cdots \\ \phi_{J,k}[0] & \cdots & \phi_{J,k}[N-1] \end{bmatrix} \begin{bmatrix} x[0] \\ \cdots \\ x[N-1] \end{bmatrix} \tag{20}$$

Here, the matrix elements are the sampled version of signals $\psi_{j,k}(t)$ and $\phi_{j,k}(t)$, with each signal becomes a

row of the matrix. If the length of the samples is N, the matrix is $N \times N$, and the complexity becomes $O(N^2)$.

To reduce the DWT complexity, a ***pyramidal algorithm*** can be used based on the multiresolution decomposition explained before. In the space V_j, $f(t)$ can be represented by

$$d_{j,k} = \langle f(t), \phi_{j,k}(t) \rangle \equiv \int_{-\infty}^{\infty} f(t)\, \phi_{j,k}(t) dt \qquad (21)$$

Since each $\phi_{j,k}(t)$ is a member of both V_j and V_{j-1}, while the set of $\phi_{j-1,k}(t)$ is an orthonormal basis of V_{j-1}, *we can express any* $\phi_{j,k}(t)$ as

$$\phi_{j,k}(t) = \sum_l \langle \phi_{j,k}(t), \phi_{j-1,l}(t) \rangle \phi_{j-1,l}(t) \qquad (22)$$

By changing the integration variable in the inner-product, it is easy to show that there exists $h[n]$ defined as

$$h[n] \equiv \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \phi(\tfrac{1}{2}t)\phi(t-n) dt \qquad (23)$$

such that

$$\langle \phi_{j,k}(t), \phi_{j-1,l}(t) \rangle = h[l-2k] \qquad (24)$$

Thus Eq. (20) becomes

$$\phi_{j,k}(t) = \sum_l h[l-2k]\, \phi_{j-1,l}(t) \qquad (25)$$

Combining Eq. (21) and Eq. (25), and applying inner-product properties [Fran69], we obtain

$$d_{j,k} = \sum_l h[l-2k]\, d_{j-1,l} \qquad (26)$$

This relationship is very important, because one can efficiently compute $d_{j,k}$ from the previous $d_{j-1,l}$. Thus, by defining $x[n]$ as $d_{0,n}$, one can iteratively calculate all subsequent $d_{j,k}$ for j = 1, 2, ... using Eq. (26). Observe that Eq. (26) is essentially a filtering process of $d_{j-1,l}$ using a non-recursive filter of impulse responses $h[n]$, followed by subsampling by two. If we call this operation as H. Then

$$d_{j,k} = H \cdot d_{j-1,l} \equiv \sum_l h[l-2k]\, d_{j-1,l} \qquad (27)$$

We can arrive with a similar relation for $c_{j,k}$, since $\psi_{j,k}(t)$ is a member Of both W_j and V_{j-1}, while the set of $\phi_{j-1,k}(t)$ is an orthonormal basis of V_{j-1}. Using similar derivation yields

$$g[n] \equiv \frac{1}{\sqrt{2}} \int_{-\infty}^{\infty} \psi(\tfrac{1}{2}t)\phi(t-n) dt \qquad (28)$$

and then, after defining an operator G,

$$c_{j,k} = G \cdot d_{j-1,l} \equiv \sum_l g[l-2k]\, d_{j-1,l} \qquad (29)$$

Both Eqs. (27) and (29) replace Eqs. (17b) and (17a), respectively.

Thus in this pyramidal algorithm, the DWT becomes a processing of input signal $x[n]$ through a pyramid of operators H and G for j = 1 to J. In each stage of j, the outputs of the operator G are collected as $c_{j,k}$, while the outputs of the operator H are reapplied as inputs for the next stage.

Although the above derivation is for the forward DWT, similar approach can be used to derive 'upside-down' pyramidal algorithm for the inverse of DWT. In fact, the derivation results in a simple relation between the lower stage to its next upper stage as follows

$$d_{j-1,l} = H^* d_{j,k} + G^* c_{j,k} \qquad (30)$$

where H^* and G^* are the adjoints of H and G [Daub88].

To show that this algorithm is efficient, consider a DWT of $x[n]$ having N samples. Suppose that the total of computational cost of one input sample in a stage is a constant c. At the first stage, the cost is clearly NC. At the second stage, there are only N/2 samples to be processed due to the subsampling process, resulting in a cost of $Nc/2$. Similarly, the third stage requires $Nc/4$. This process can continue until there is no input available for the next stage. Thus total complexity is

$$Nc\left(1 + \frac{1}{2} + \frac{1}{4} + \ldots\right) \le 2Nc \qquad (31)$$

which is proportional to NC instead of N^2. We can estimate the c if we know the length of $h[n]$ and $g[n]$. Suppose the lengths of $h[n]$ and $g[n]$ *are the same,* which is $2L$, for an $L \in Z$. Then NC must be the number of multiply-and-accumulate (MAC) processes to complete both convolutions in Eq. (17) and Eq. (19) for all input samples at the first stage, which must be equal to $2LN$. Thus, total MAC processes to complete the DWT is $4LN$, as opposed to N^2. Table 1 shows the complexity comparison with and without pyramidal algorithm.

Table 1. Comparison of complexity of DWT and pyramidal DWT, for N input sample and filters of length *2L*.

Algorithm	*O*()	Complexity *N*=64, *L*=2
Basis Inner Product	N^2	*4096*
Pyramidal Algorithm	$4LN$	512

2.6. Daubechies Wavelets

Ingrid Daubechies has derived a class of compactly supported wavelets that are compatible with Mallat's multiresolution analysis and pyramidal algorithm. Forcing the wavelets to be compactly supported while maintaining the compatibility, she obtained a method of constructing such wavelets, as well as some properties [Daub88], such as

$$\sum_n h[n] = \sqrt{2}$$
$$\sum_n g[n] = 0 \qquad (32)$$
$$g[n] = (-1)^n h[2m+1-n] \qquad m \in Z$$

The simplest Daubechies wavelet is the DAUB4, with *L* in Table 1 is two. The filter part of *H* has four impulse responses *h*[0], h[l], *h*[2], and *h*[3], with values

$$\begin{aligned} h[0] &= 0.4829629\ 1 \\ h[1] &= 0.8365\ 163 \\ h[2] &= 0.22414387 \\ h[3] &= -0.12940952 \end{aligned} \qquad (33)$$

The filter part of G also has impulse responses *g*[0], g[l], *g*[2], and *g*[3] which are related to *h[n]* according to Eq. (32). Here, we select *m*=1, such that *h*[*n*] and *g[n]* have a similar filter structure (down to tap positions), simplifying the implementation. This option of *m* also results in wavelet and scaling prototypes of the same supports.

2.7. Extending to Two-Dimensional Signals

A scheme to extend the DAUB4 DWT for two-dimensional signals is similar to that of the Fourier transform. Here, the computation is in two steps [Pres9 la]. First, we apply a one-dimensional DWT sequentially on the columns and replace each column of the image with its DWT results. Second, we redo the similar transformation, but now on the rows, resulting in two-dimensional wavelet representation. Since the transform is orthogonal, the number of representation data is the same with the number of pixel in the original image.

3. WAVELET COMPRESSION

3.1. Basic Scheme

Lossless compression works by identifying redundant data and removing them [Kins91]. The redundancy may come from non uniform distribution of data as well as data correlation. If more compression is still necessary, the scheme further looks for irrelevant data, converts them into redundant data, and then removes them. The compression then becomes lossy because the irrelevant data cannot be recovered.

The irrelevancy can be defined according to different criteria, ranging from subjective fidelity criteria to objective fidelity criteria [GoWi87]. Irrelevant data may be those with little contribution to the fidelity. They may also be those containing little information in the Shannon information theory sense. Clearly, one can associate a measure on irrelevancy, reflected in a quality measure.

Compression methods must then concentrate on redundancy removal and irrelevancy reduction [JaJS93]. Redundancy removal methods include predictors and transforms, while irrelevancy reduction methods include quantization and data elimination. They are often combined. For example, quantization may take place in the transform domain and/or predictor error.

In wavelet approach, it is possible to employ both predictors and transforms. The wavelet transform results carry both time and frequency samples that may have sample correlation, thus predictors can be used. Furthermore, the transform itself may have removed the redundancy and made the irrelevant data more visible in the wavelet domain. We thus study some of the possibilities through experimentation. Especially, we try to find parameters that govern the irrelevancy and redundancy.

3.2. Experimental Results

In this experiment, we study the role of the wavelet coefficient *magnitude* in governing irrelevancy. We setup the experiment to study its effect on quality (related to irrelevancy) as well as bit rate (related to redundancy). Based on a scheme shown in Fig. 2., the scheme first takes the DWT of the original image. As the primary test data, we have selected `lena . img.` A less intensive experiment was performed also on `camera. img` for confirmation (see Fig. 3(a) and Fig. 4(a)). The inverse DWT can convert the image back from wavelet domain to the spatial. domain for viewing. We can then perform various processes on the transformed data. By observing the processing effects on the quality

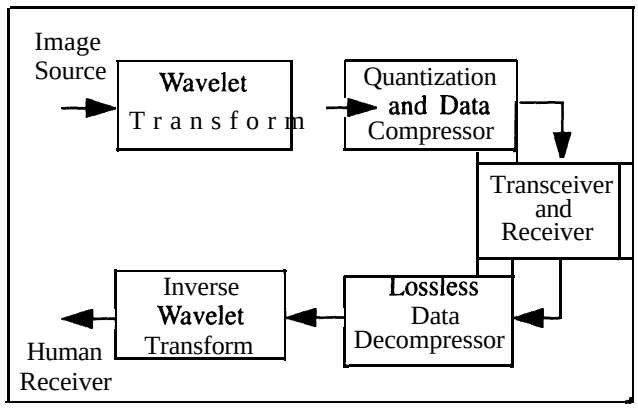


Fig. 2. A compression scheme using wavelets.

as well as the bit rate, we can design a compression scheme.

To facilitate the quality measurement, a routine computes the peak signal-to-noise ratio (PSNR) according to (in dB)

$$PSNR = 10\log\frac{255^2}{MSE} \tag{34}$$

The mean square error (MSE) is the average of the energy of the difference between the original and the reconstructed images. We loosely define good quality as having a PSNR of greater than 30 dB.

One way to study the magnitude effect is through a coefficient truncation scheme. Here, we use a truncation threshold, such that coefficients with absolute values smaller than the threshold are considered irrelevant, and converted into zero to become redundant data. Severe truncation leads to data losses, but resulting in an efficient compression. Thus, we study the loss in PSNR and subjective quality of the image, while at the same time observe the reduction in the size of the image code file.

Our observation reveals that the smaller the magnitude of the coefficients, the more irrelevant the coefficients are. Increasing the threshold results in more coefficients being truncated. Figures 3(b) to 3(f) and Table 2 show the effects of various degrees of truncation to the image quality. The truncation results in good quality at up to 90% truncation. As shown in Fig. 5, there are only few coefficients that are responsible for total fidelity. Those coefficients must have high coefficient values, because they survive the truncation.

One explanation is that the wavelet transform is an orthonormal transform following the principles of Eq. (1) and (3). Such a transform satisfies the Parseval's

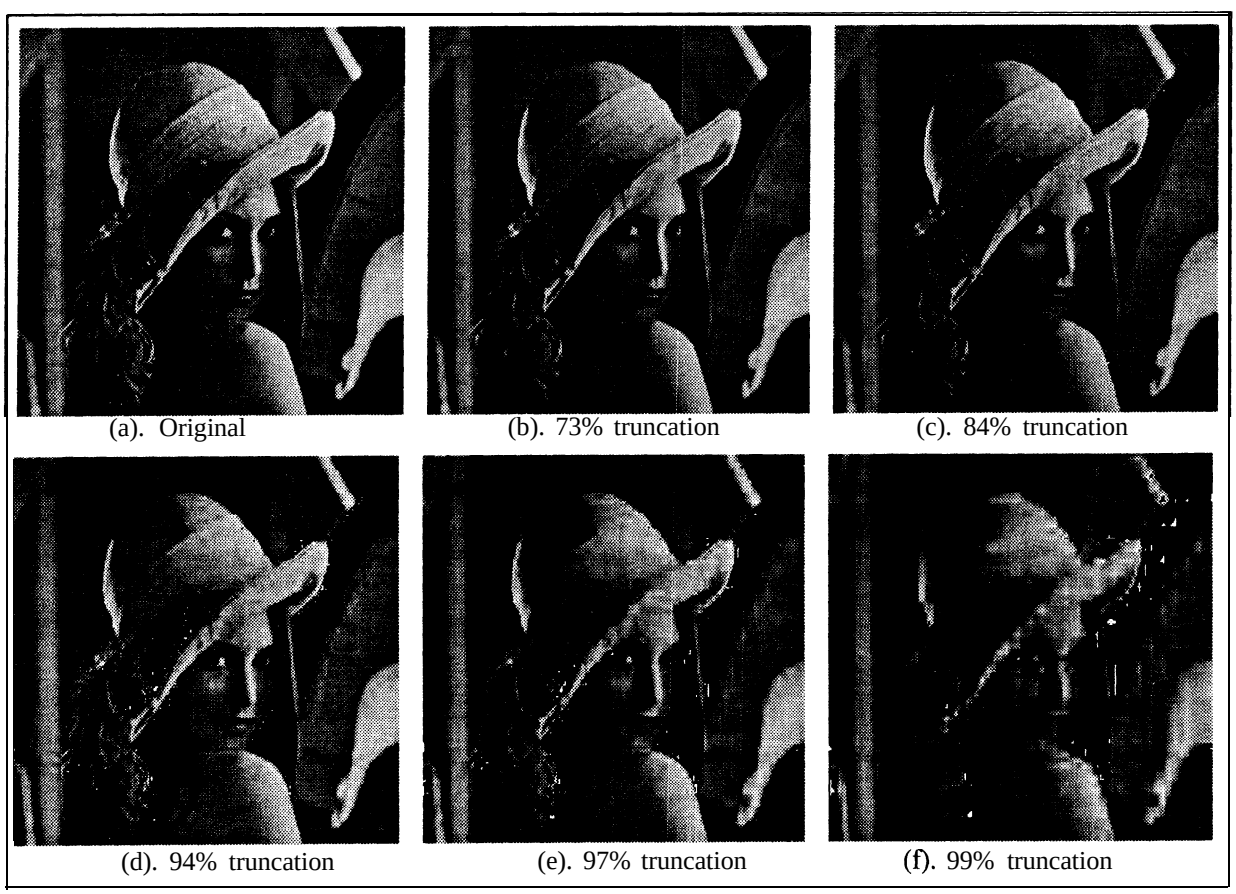
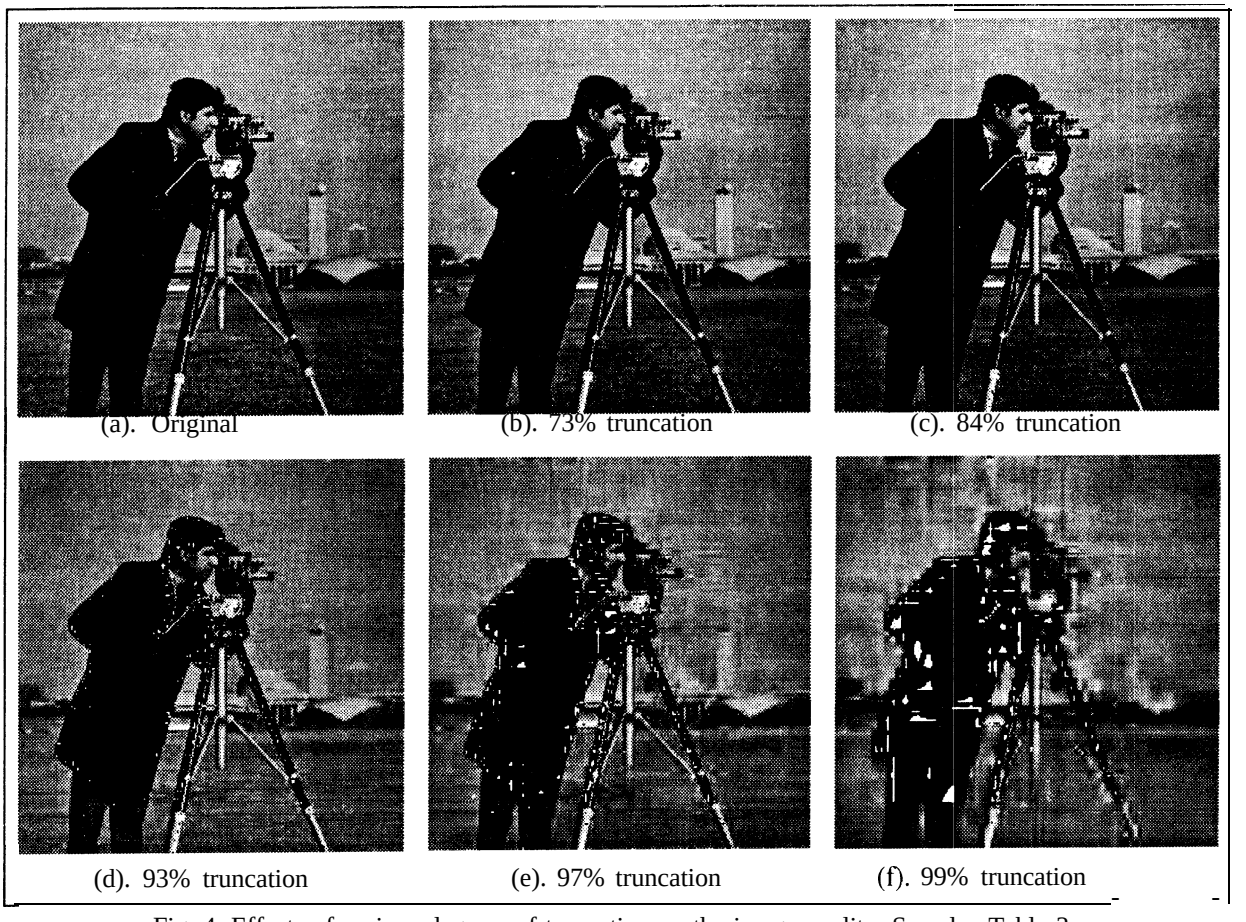


Fig. 3. Effects of various degrees of truncation on the image quality. See also Table 2.



(a). Original (b). 73% truncation (c). 84% truncation

(d). 93% truncation (e). 97% truncation (f). 99% truncation

Fig. 4. Effects of various degrees of truncation on the image quality. See also Table 3.

Table 2. The PSNR values and bit rates of the images shown in Fig. 3.

% Truncation	PSNR (dB)	Bit Rate (bpp)
77	39.16	3.8
86	33.23	2.5
94	27.16	1.4
97	23.76	0.7
99	21.08	0.36

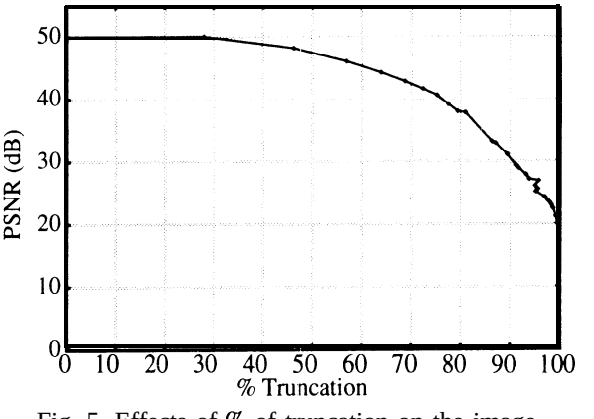


Fig. 5. Effects of % of truncation on the image quality for `lena.img`.

relation which equates both energies in the original and wavelet domains [Fran69]. Since the wavelet basis is orthonormal, the greater the magnitude of one coefficient, the higher its contribution to the energy. Thus truncating the such coefficients would result in high MSE values, reducing the PSNR.

For more truncation, the distortion is localized and looks like salt-and-pepper noise, as shown in Fig.3(b) to 3(e). This is due to the fact that the coefficients responsible for the constructing the pixels have been eliminated. This also verifies one of wavelet properties of being localized in both spatial and frequency domain. Thus quantization error in the wavelet domain does not translate in distortion distrilbuted all over the image. Low pass or median filters should be able to reduce the effects of such a distortion. Severe truncations still pre-

serve the general looks, as shown in Fig. 3(f). The images are very distorted, but the general feature is still observable.

The set of truncated coefficients has an unbalanced distribution of zeros now such that a **lossless** compression should be able to compress it. We can then use the **ZIP** compression to compress the coefficients. As shown in Fig. **6, 90%** of truncation translates to 2 bpp representation. The ZIP program was able to identify data repetition in the truncated coefficients, and encoded them compactly.

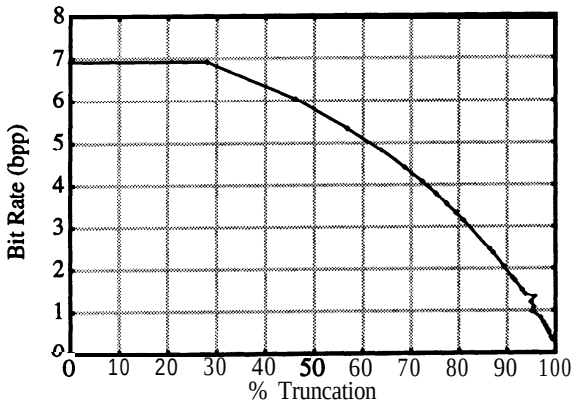


Fig. 6. Effects of the truncation to ZIP compression for `lena.img`.

The same scheme is also used on another image, `camera. img`, with almost similar results. Although the quality of the reconstructed image is not as good as that of `lena. img`, the results show similar trends (see Fig. 7 and 8). Also, Fig. 4 and Table 3 show similar effects of the truncation on the image quality.

Table 3. PSNR values and bit rates of images in Fig. 6.

% Truncation	PSNR (dB)	Bit Rate (bpp)
73	40.18	3.98
84	31.10	2.8
93	23.26	1.6
97	19.51	0.85
99	17.46	0.39

3.3. Compression Performance

We can then use the threshold scheme as a simple image compression scheme. The compression takes the forward **DWT** of the image, truncates the small magnitude coefficients according to a threshold, and losslessly compresses the truncated coefficients to become the

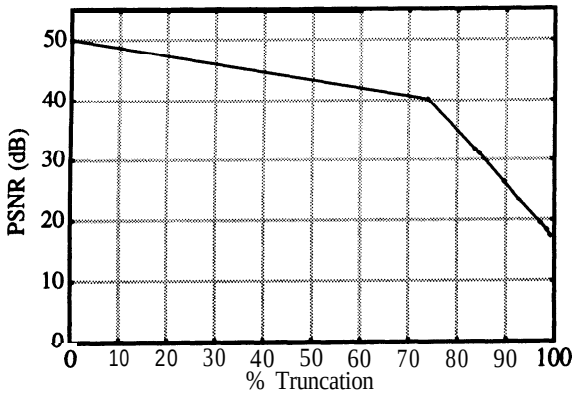


Fig. 7. Effects of % of truncation on the image quality for `camera. img`.

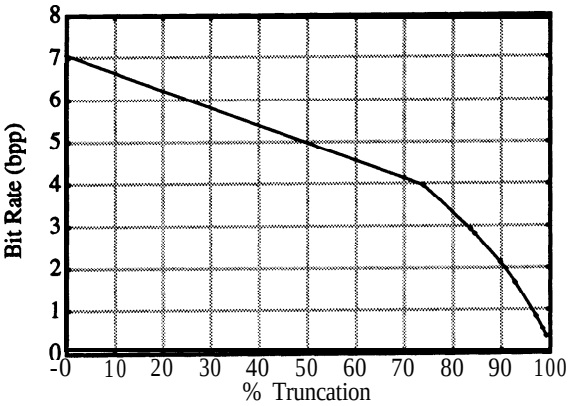


Fig. 8. Effect of the truncation to **ZIP** compression for `camera.img`.

compressed image. The decompression then starts with **lossless** decompression of the compressed image, and inverse transforms the results to obtain **viewable** image. Figures 9 and 10 show the compression performance measured for `lena. img` and `camera.img`, respectively. We observe that in 2 bpp the image still have good quality.

4. Discussion

Localization characteristic and sparsity of **wavelet** representation are interesting features for image compression. The localized property of the **wavelet** coefficients reduces the effect of a quantization **error** to the total image. This is an advantage over Fourier representation. The **sparsity** means that most of the image energy is distributed among a few basis functions only. Thus a scheme that finely quantizes the high coefficients while coarsely quantizes the small-valued coefficients leads to efficient compression with high quality.

From a practical perspective, the compression scheme is interesting due to the simple and fast computation structure. Since the implementation of the **wavelet**

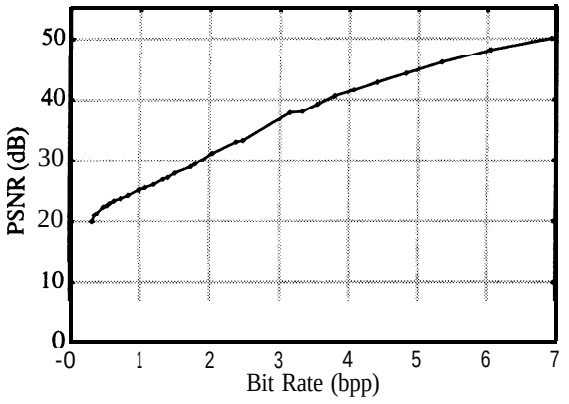


Fig. 9. Quality of the image for different compression rates for `lena.img`.

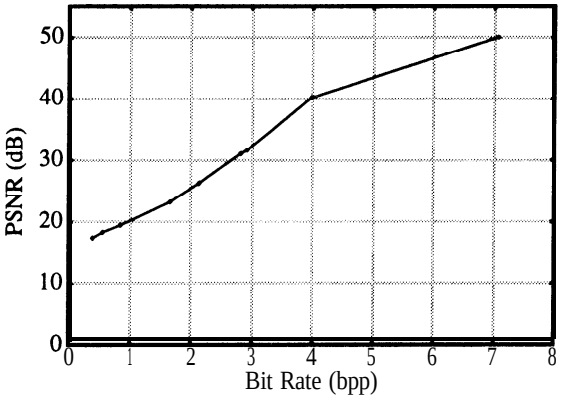


Fig. 10. Quality of the image for different compression rates for `camera.img`.

transform pair consists of filtering only, fast dedicated hardware is possible.

In this scheme, two aspects are open for improvements: (i) selection of the wavelet prototype, and (ii) compression of the coefficients. In our experiment, we selected DAUB4 because of its simplicity, its localized property, and its immediate availability. Thus, the selection has not been made through a study of characteristics of different wavelets. A more elaborate choice of wavelet may lead to much better results. For example, [Mall89] reports the use of image distribution characteristics into consideration results in 1.5 bit/pixel, with only a few noticeable distortions. Furthermore, in our experiment we apply brute force truncation to eliminate redundancy. A more thoughtful method should produce better results. A program which optimally embeds Huffman coding in the compression scheme results in 3: 1 lossless compression and 50: 1 lossy compression with some degradation [Pres9 1 b].

We conclude that wavelet coding is an important method for image compression. Characteristics of the wavelet representation such as locality and sparsity can be exploited for compact representation. The magnitude of a wavelet coefficient determines the coefficient's significance with respect to the image fidelity. This can lead to very promising compression schemes as demonstrated in this paper.

ACKNOWLEDGMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the Telecommunications Research Laboratories (TRLabs). One of the authors (AL) wishes to thank ILK-Microelectronics ITB, Laboratory of Signals and Systems ITB, and PT INTI Persero, Bandung, Indonesia, for their support in this research work.

REFERIENCES

[AnRA91] P. H. Ang, T. A. Ruetz, and D. Auld, "Video compression makes big: gains", *IEEE Spectrum Magazine*, IEEE Cat. 0018-9235/91, pp. 16-19, Oct. 1991.

[Daub88] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Comm. Pure. App. Math., v.* XLI, 1988, pp. 909-996.

[Fran69] L. E. Franks, *Signal Theory. New* Jersey: Prentice-Hall Inc., 317 pp., 1969.

[GoWi87] R. C. Gonzales and P. Wintz, *Digital Image Processing.* Reading MS: Addison-Wesley, 502 pp., 1987

[JaJS93] N. Jayant, J. Johnston, and R. Safranek, "Signal compression based on models of human perception," *Proceeding IEEE*, 1993, v. 8 1, no 10, pp. 1385-1421.

[Kins91] W. Kinsner, "Review of data compression methods, including Shannon-Fano, Huffman, arithmetic, Storer, Lempel-Ziv-Welch, fractal, neural network, and wavelet algorithms", *Technical Report*, DEL91-1, University of Manitoba, 157 pp., Jan. 1991.

[Mall89] S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no 7, pp. 84-95, July 1989.

[Prag92] D. S. Prague, "How will multimedia change system storage ?", *BYTE Magazine*, McGraw-Hill, Peterborough NH, pp. 164-165, Mar. 1992.

[Pres91a] W. H. Press, Wavelet Transforms, *Harvard-Smithsonian Centre for Astrophysics Preprint No.* 3184, 1991. (Available through anonymous ftp at 128.103.40.79, directory `/pub`).

[Pres9 1 b] W. H. Press, FITSPRESS, *Harvard-Smithsonian Centre for Astrophysics*, Beta Version 0.8, 199 1. (Available through anonymous ftp at 128.103.40.79, directory `/pub` and file `fitspress08.tar.z`).

[RiVe91] 0. Rioul and M. Vetterly, "Wavelets and signal processing", *IEEE Signal Processing Magz.*, IEEE CT 1053-5888/91 pp. 14-38, Oct. 1991.

ROSE X.25 Packet Switch Status Update

by
Thomas A. Moulton, W2VY
Radio Amateur Telecommunications Society

INTRODUCTION

The past year has been very busy for both the development and the network expansion. Highlights include support for 64k EPROM, reformatted applications, X.29 Invitation to Clear and enhancements for IP users.

These changes and others that are planned continue to expand the richness of the ROSE X.25 Packet Switch as a network backbone tool. It can pass data transparently with no changes being made to the applications that people are using.

BUG FIXES

The following is a list of bugs or symptoms that have been fixed:

- When first installing the ROSE EPROM, the TNC needs to be power cycled twice to come alive.
- Connections made using DIG1 Callsign were not being made using the DIG1 Callsign of the remote node (TCP/IP Users).
- Frame boundaries were not preserved for Non-AX25 Level 2 User frames (TCP/IP Users).

APPLICATIONS

The applications have had many changes in the basic way they operate. Each application now has a prompt that identifies the application and network address you are connected to. They also support a Bye command which will cause the switch to disconnect from you. Some applications such as INFO and SHEARD will just send their data and disconnect. They will now support all SSID's. This means that you can connect to a specific application in more than one switch at the same time. For example you could connect to USERS-l and USERS-2 to access the USERS application at two different switches to monitor the path a call request takes.

The USERS application now displays the network address of it's neighbor nodes. This will let users walk through the network to learn the topology and to discover new user channels.

The CONFIG application now supports a disconnect command (:0300000000). It also now supports a password challenge which must be entered correctly to obtain write access to the configuration information. If the password is not entered correctly the user will have read only access to the configuration interface.

CONFIGURATION OPTIONS

The switches now support full 14 digit X. 121 addressing. You may configure, if required, a LID LIST to bar access by a station. The frame and packet level timers are now defined in 100ms intervals to allow for better tuning of 9600 baud or greater trunk speeds.

TCP/IP

Two of the three bugs that were fixed directly impact TCP/IP users. With these fixes IP Users will be able to send a Datagram of *ANY* size using the fragmentation provided in NOS. There should be no problems with sending 1K, 2K, 4K, or even 10K datagram in 256 byte or smaller frames. These enhancements should greatly improve the performance of IP connections through ROSE and should provide the best performance.

64K EPROM SUPPORT

The ROSE X.25 Packet Switch now offers an option to install a 64K EPROM which provides the normal applications in EPROM. The applications that are included are: USERS, INFO(English), HEARD, CONFIG.

The larger EPROM is supported on the TNC-2 Clones, Tiny-2, Sprint-2 and TNC-320 and DR-200.

FUTURE ENHANCEMENTS

TXUI - Application to pass UI frames through the network based upon the network address in the digipeater fields. The digipeater fields are formatted much like normal ROSE connections, switch call and address. Users will be able to define a list of which remote switches the UI frames should be sent to. This will allow for sending a CQ to all User channels in a given region or local handling for "Mail For:" beacons and distribution of ARP frames to all IP channels.

NODE - A TheNet NODE style interface that will allow you to connect to a NODE application in a switch and then use that as a launch point to connect to another switch or a user. It is important to note that the user's callsign will still traverse the network without any changes.

CONCLUSION

The ROSE X.25 Packet Switch has undergone a significant number of enhancements that set it apart from other networking schemes. It offers flexibility for both users and system managers while simplifying the connection setup process for all.

There have been discussions with groups that are using TheNet about using ROSE in the highest level backbone to increase the geographic region that the nodes can reach. Between the transparent PID support and TXUI broadcasting we could very well convert most of, if not all, the backbone to ROSE and still support TheNet as a network application.

The goal of ROSE is to provide powerful tools to enhance communications independent of the actual protocols being used.

A primer on reliability as applied to amateur radio packet networks.

T.C. McDermott, N5EG
Texas Packet Radio Society, Inc.

1.0 Scope

Many messages have been sent regarding linking of large number of packet radio switches, nodes, digipeaters, etc. And some have commented on the desirability of very long packet networks. This monograph will describe how to calculate the availability of such a system, given knowledge of the performance of the equipment.

2.0 Definitions

Let's define some terms, first. There are 3 basic parameters that need to be known in order to make suitable calculations about network availability.

MTBF = Mean Time Between Failures. This is the average (mean) time between failures of a particular piece of equipment. For example: an MTBF of 1680 hours would equate to a piece of equipment failing once every 10 weeks, on average.

MTTR = Mean Time To Restore. This is the average (mean) time to restore a failed piece of equipment to service. For example: a piece of equipment with an MTTR of 8 hours implies that it takes 8 hours to: 1) notice that there is a problem, 2) diagnose the problem, 3) drive to the site, 4) repair the equipment, and 5) place it back in service. Of course the actual series of events, and the time to restore all depend on whether the equipment is accessible at any time, backup equipment is available, etc.

A = Availability. This is the portion of time that a piece of equipment (on average) is available for use. This can be calculated as follows:

$$A=MTBF/(MTBF+MTTR) \quad (1)$$

3.0 Some basic probability

Let's use some of the basic rules of probability to derive the availability of networks of equipment that each has availability, A. There are two basic configurations of multiple pieces of equipment: 1) series, and 2) parallel. By this is meant the following: two pieces of equipment are in series if both are required to be operating correctly in order to get the job done. For example: suppose that you wanted to transmit a packet message across a 100-mile path, and there were two switches in the middle that were linked, and that the failure of either would prevent your packets from traversing the path.

Then those switches, from a reliability point of view, are in series. The failure of either one of them would make the path not usable. In contrast, two pieces of equipment are in parallel when either alone is capable of getting the job done. For example: suppose that you wished to send a packet between two points that are 50 miles apart, and you had a choice of either of two switches, each of which alone was capable of making the path. Then those switches, from a reliability point of view, are in parallel.

3.1 Availability of things in series

We can calculate the availability of 'n' items, all with the same availability, 'A', that are in series. The combined availability is:

$$A_{n(ser)} = A^n \quad \text{(A raised to the 'n'-th power)} \qquad (2)$$

For example, suppose that we have a packet network consisting of 20 nodes, that each individual node has an MTBF of 4368 hours (6 months), and an MTTR of 168 hours (1 week). Then the availability of a single node is 4368 / (4368 + 168) = 0.963 (96.3 percent of the time it works). The availability of a network of 20 of these nodes would be: $0.963^{20} = 0.470$ (47.0 percent of the time it works). We can see that, in general, putting items in series degrades the availability.

3.2 Availability of things in parallel

We can calculate the availability of 'n' items, all with the same availability, 'A', that are in parallel. The combined availability is:

$$A_{n(par)} = 1 - ((1 - A)^n) \qquad (3)$$

For example. suppose that we have a packet network consisting of 2 nodes, with MTBF = 4368 hours, and MTTR = 168 hours, and these two nodes are in parallel. Then the individual availability is 0.963 (as in 3.1 above) and the combined availability is 1 - (l-0.963)* = 0.9986 (99.86 percent of the time it works). We can see that, in general, putting items in parallel improves the availability.

3.3 More complex models

We can calculate the availability of more complex networks many times by reducing them to series and parallel combinations that we now know how to handle. Sometimes, the combinations are not reducible to series-parallel combinations, but these cases are not common in amateur packet networks. The general procedure is to break up a network into subsections that can be described as being in series or parallel. Then each subsection can in turn be broken up into smaller subsections that are in parallel and or series, until the remaining network segments are entirely parallel or series.

The availability of each subsection can be computed, and the subsection availability's can be combined using (2) and (3) above to derive the network availability.

4.0 Some examples

Lets look at two example networks. Network one consists of 40 packet switches, all in series. It's a long haul network, and skinny (i.e.: no alternative routes exist within the network from end-to-end). Each node has an MTBF of 4368 hours, and the MTTR is 332 hours (2 weeks, since the sysop left on vacation yesterday, and does so frequently! - nice work if you can get it). Then:

$$A = 4368 / (4368+332) = 0.929$$

$$A_n = 0.929^{40} = 0.053$$

This network will function, end-to-end, 5 percent of the time, and will not work end-to-end 95 percent of the time. Hmmmm. OK, let's assume that our sysop loses his cushy job, his extravagant vacation policy, etc., and can get to the site within 72 hours. Then the network availability would be:

$$A = 4368 / (4368+72) = 0.984$$

$$An = 0.984^{40} = 0.520$$

Well, quite an improvement. The network actually works, end-to-end, 52% of the time. We can draw some conclusions about the level of service our poor sysops are going to have to provide if we want this stuff to really work. Alternatively, we could do some work up front, and build dual-redundant nodes. Those are ones with hot-standby equipment that takes over the failed equipment with no loss of service (even after lightning hits!). So, for network example number two, let's double the investment in our network by providing dual-redundant nodes at each of the 40 sites. Incidentally, building dual-redundant equipment without common (joint) failures can be no small task in itself. The availability of this network, assuming our intrepid sysop finds out that he now takes 2 week business trips all the time, can be calculated by breaking our network into some subsections. Each dual-redundant node is a subsection, and we have 40 of those subsections in series. So, first we calculate the availability of the subsection:

$$A_{2(par)} = (1 - (1 - 0.929)^2) = 0.995$$

and then the availability of all subsections would be:

$$A_{40(ser)} = 0.995^{40} = 0.817$$

Well, that's more like it. This network works 81.7 percent of the time, end-to-end, and the poor sysop can actually hold down a real job now. Ahhh, wait a minute. We have twice as much equipment in the network now, and thus it seems like twice as many things would break. Well, yes. Welcome to the dark side of the force - err, the dark side of high availability. In order to achieve this level of availability, we have to fix any failed equipment within 2 weeks - even if the failure does not take the node out of service. If we don't fix it, then the remaining part that still works now

determines the node's availability, and we are no better off than before (at this node). So, there is no free lunch. Also, we have to be able to detect that something at the node has failed, even though it is still working. OK, so let's just put up two different packet networks each one of which reaches the two endpoints, but without any of this dual-redundant nonsense. In this case, we can model the two subsections as 40-element series connections of non-redundant nodes, and then we have two of these long strings in parallel.

$A_{40(ser)}$ = 0.053 (from above)

$A_{2(par)} = (1 - (1 - 0.053)^2) = 0.103$

Well, this strategy didn't work very well compared to making each node dual-redundant. So it seems like our poor sysop is stuck in engineering dual-redundant nodes if we want our networks to work reliably. Commercial telecom equipment is generally engineered this way.

5.0 Conclusions

Some conclusions we can draw about the results from our two examples are:

1) Engineering of fault-tolerant nodes is essential for long packet routes based on a number of packet nodes. Redundancy can be provided at the equipment level, or with alternate nodes having the same connectivity to the network.

2) Or, alternatively, we should focus on long-hop technologies such as satellite, HF, scatter or land-line telephone(?) connections. One drawback of such long-haul technologies is that we may lose real-time communications between end operators. Usually, gateway stations perform store-and-forward routing over HF, satellite, and land-line connections. And effective scatter communications probably would require specialized high-power gateway stations also. These techniques usually lose real-time capability since he media may not support propagation 100% of the time (e.g.: HF).

FSK modem with scalable baud rate

Wolf-Henning Rech, N1EOW, DF9IC @ DB0GV.DEU.EU, Pariser Gasse 2, D-64347 Griesheim, Germany
Gunter Jost, DK7WJ @ DBGV.DEU.EU, Lichtenbergstr. 77, D-64289 Darmstadt, Germany

Introduction

Binary FSK modulation of the RF carrier is a well known and widely used transmission scheme for digital information. Old **RTTY** modems are the **first** example in amateur radio; nowadays packet radio is the most prominent FSK application.

The advantage of FSK is compatibility with existing FM voice radios and simple implementation, including robustness against misalignment, bad transmission characteristics of the radios, or **frequency** drift. More sophisticated RF modulation schemes are rarely used until now. Combined with coding techniques they could provide more efficient use of the RF bandwidth at low SNR but need very careful construction of both modem and transceiver. Subcarrier modulation techniques, like AFSK or AQPSK, are disadvantegous in respect to RF bandwidth and SNR.

Even succesfull digital cellular phone systems like the European GSM use GMSK (Gaussian minimum shift keying) which is basically FSK with controlled deviation and a baseband partial response **prefilter.** For this reason we believe that FSK will be used in packet radio for another long time.

Well-known amateur radio FSK modem concepts are those of **K9NG** and **G3RUH** which both use an additional scrambler with the polynomial $1+x^{12}+x^{17}$. Their implementations are different in detail but basically compatible at the air interface. **G3RUHs** filtering is more sophisticated with an EPROM hard-coded FIR filter in the transmitter to achieve best spectral purity without alignment.

Both are designed for 9.6 kbaud, need an RF bandwidth of about 20 **kHz** and work fine with IF filters down to **15 kHz.** They can be modified to other baud rates by changing a large number of parts in the analog filters and using another clock frequency. This can be useful to double the speed to 19.2 kbaud with slightly modified radios with 30 **kHz** IF bandwidth, or even more with broader **IFs,** like broadcast type **WBFM** circuits.

We have improved the **G3RUH** modem keeping in mind the idea of simple baud rate scalability through the clock frequency **[1,2].** This task can be solved by replacing the analog filters through switched-capacitor circuits combined with a completely digital DCD circuit. To care for birdies, broadband noise and other 'alias' effects of time discrete filter circuits some analog anti-aliasing filters are added which restrict the scalability to a range of **1:8** in baud rate, e.g. from **9k6** to **76k8.**

The use of **PLDs** in the circuit allow a compact design and a reduced part count. Additional functions like a bit regenerator for repeater-like 'data echo' operation at nodes are already included.

Hardware description

The modem circuit is shown in Fig. 1 (analog) and Fig. 2 (digital). These drawings are part of a larger circuit diagram of a complete node controller with the modem on-board so that the modem part numbering is not continuous, The circuit operates from 5V single supply. The operating current depends mainly on the type of the GALs used.

The interface to the node controller uses a baud rate x32 clock delivered by the SCC or equivalent which allows software control over the baud rate. The circuit assumes that the final RX clock is produced by the HDLC controller - if not it can be used from pin 19 of IC18 after inversion. The other lines are as usual (TxD, RxD, DCD, RTS, CTS). RxD and TxD must be NRZI coded. DTR has a special meaning (cf. next chapter).

IC13/14 form the TX scrambler; 8 outputs are connected to IC15 which contains the TX FIR lowpass filter. Its output data is DA converted by IC16 (output TFsk in Fig. 3). This part of the circuit is equivalent to G3RUHs transmitter but with less parts. It is important to note that the content of the EPROM IC15 is completely different from G3RUHs.

The analog TX filter is shown in the lower part of Fig. 1. The TFsk-Signal is fed to the input of IC27 which is a 8-pole Butterworth lowpass filter in switched-capacitor technique. Its comer frequency is 1/50 of the clock. We use the baud rate x32 clock to achieve a ratio of 0.64 between filter cut-off frequency and baud rate which is close to optimum.

The SC filter output is the filtered again through a 3-pole fixed frequency continuous lowpass realized by the additional opamp in IC27 and IC28B. It is dimensioned for the range 4k8-38k4, as shown, but the capacitors can be changed. The levels and gain of the stages has to be carefully chosen to avoid clipping at the low supply voltage.

Output connector, loopback jumper and other parts are not drawn in the diagram because they are application specific.

The receiver begins with a simple 2-pole fixed frequency continuous lowpass followed by another 8-pole SC filter. IC26 has Bessel characteristics resulting in minimal overshot. The high order results nevertheless in a good noise suppression. The output is threshold compared by IC28A; R15 introduces a weak hysteresis to improve stability.

IC18 (Fig. 2) samples the comparator output with the x32 “microclock”. It contains a DPLL and generates the RX clock plus a raw DCD information (DPLL locked/unlocked). IC22 and parts of IC21 form the descrambler while the other ICs serve for DCD processing.

JP4 allows the selection of a bit regenerating function to retransmit received signals on duplex user access points for monitoring purposes and collision avoidance. RxD is switched to TxD and PTT is activated when DCD comes up.

JP5-8 select the filter function as usual. 16 banks are available - 8 of them are filled with code in the moment.

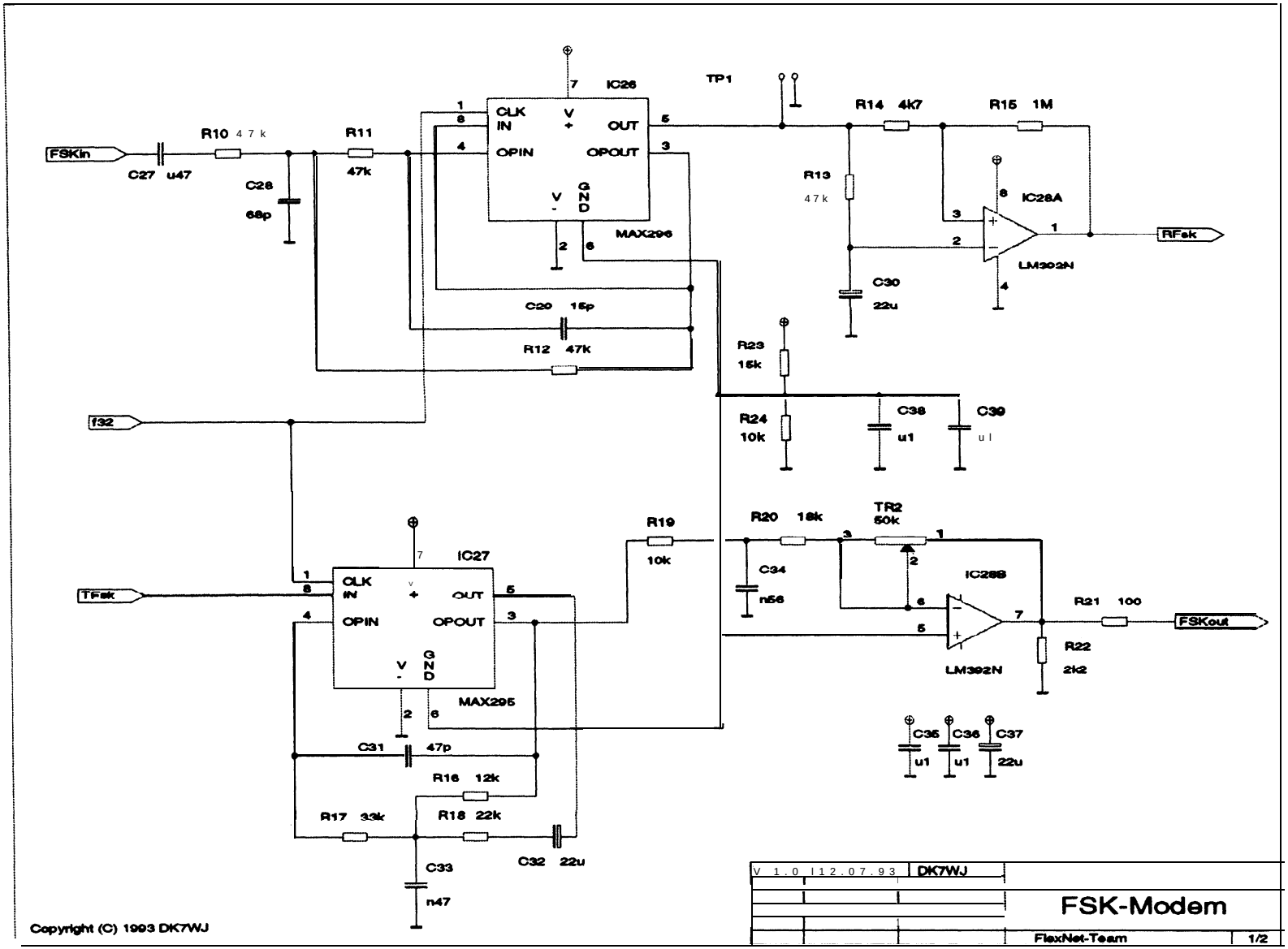


Fig. 1 Analog circuit

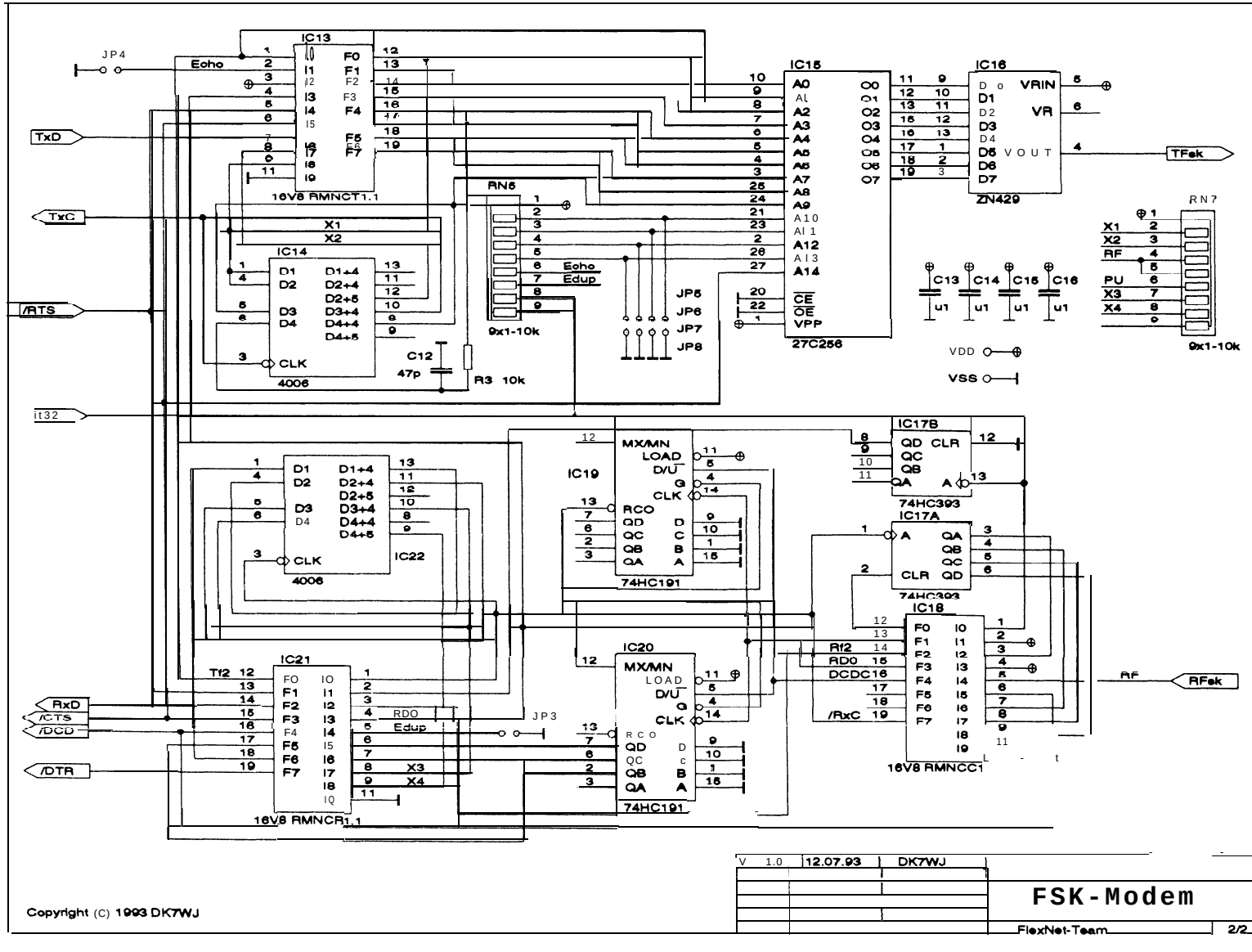


Fig. 2 Digital circuit

Filter design, DCD, and bit regenerator

There is a large number of filters of different types in the circuit. The transmitter signal is generated by the DAC with fourfold oversampling, e.g. during 9k6 operation 38,400 samples per second are generated from the EPROM table.

This results in a spectrum which can be tailored to the demands within a frequency range from DC to 19,200 Hz. Above there are unavoidable spurs called “alias” products.

To care for them the SC filter operating with a cutoff frequency of 6150 Hz has reached enough attenuation until 19,200 Hz through its high order. But this filter works in a discontinuous manner with a clock frequency of 307,2 kHz in our example. Clock f&through and up-converted signals around this frequency have to be removed again. The following 3-pole continuous filter with a corner frequency of 70 kHz provides enough attenuation at 300 kHz if the baud rate is 9k6, without introducing distortion to the signal in the case of 76k8 operation.

Fig. 3 shows the output spectrum of the modem operated with 9600 Baud.

The receiver circuit is equivalent but in the vice-versa order. The continuous input filter prevents the SC input from noise or IF signals at very high frequencies. The SC filter limits then the noise bandwidth to its final value.

The code in the EPROM is different from G3RUHs TX EPROM for the following reasons:

- one input line has inverse polarity which reflects in the data arrangement
- the target waveform has been derived independently [3] and might differ very slightly
- the analog postfilter is of a higher order and another type which is compensated for in the coded samples
- instead of the measured characteristics of some arbitrary radios the predistortion of the various waveforms compensates first order low pass filters of descending comer frequency
- the output voltage range is different

The code for both the GALs and the EPROM is free for non commercial use in amateur radio - ask one of the authors for files or samples.

The DCD circuit is a quite complex part of the modem. The soft 32-stage DPLL is the base for it. A signal is derived at each zero crossing which indicates if the crossing time was “good” (synchronous to the preceeding waveform) or ‘bad” (asynchronous). The discrimination threshold is dynamically ajusted to improve DCD speed and stability, between 25% of the frame interval1 around the transition of the DPLL when DCD is actually off, and 50% of it when DCD is on.

This DCD raw signal switches an &bit-counter to count up or down. It is clocked at 4 times the baud rate so that 6 bit are effectively in use. If more “good” than “bad” events occur it counts up and reaches a threshold where DCD is activated. This threshold has again a hysteresis so that deactivating occurs at a lower count. Both hysteresis are coded in the GAL circuit.

The complete DCD circuit is digital and synchronous to the x32 microclock so that it is baud rate transparent. No (fixed) integrating time constants are needed. The DCD response time is about 50 clock signals average (5ms at 9600 Baud) depending on tlhe signal quality. Even noisy signals produce no flicker.

The bit regenerator option is activated through JP3. On DCD response it switches the **PTT** line. If JP4 is also on it switches the source for **TxD** from the node controller interface to the own **RxD** line. The **TxC** is then derived the same way from the **RxC** out of the DPLL. So the received and retimed data is sent again.

If JP3 is on and JP4 is off only the **PTT** mechanism is active. Then flags (or whatever comes out of the HDLC circuit) are sent. In both modes the node controller has priority over the data or flag echo.

In addition this echo function can be enabled or disabled **from** the node controller through the DTR line. This allows remote control over the transmitter in case of interference (otherwise distant users would key the transmitter each time when the node receives their signal even if the node itself is off).

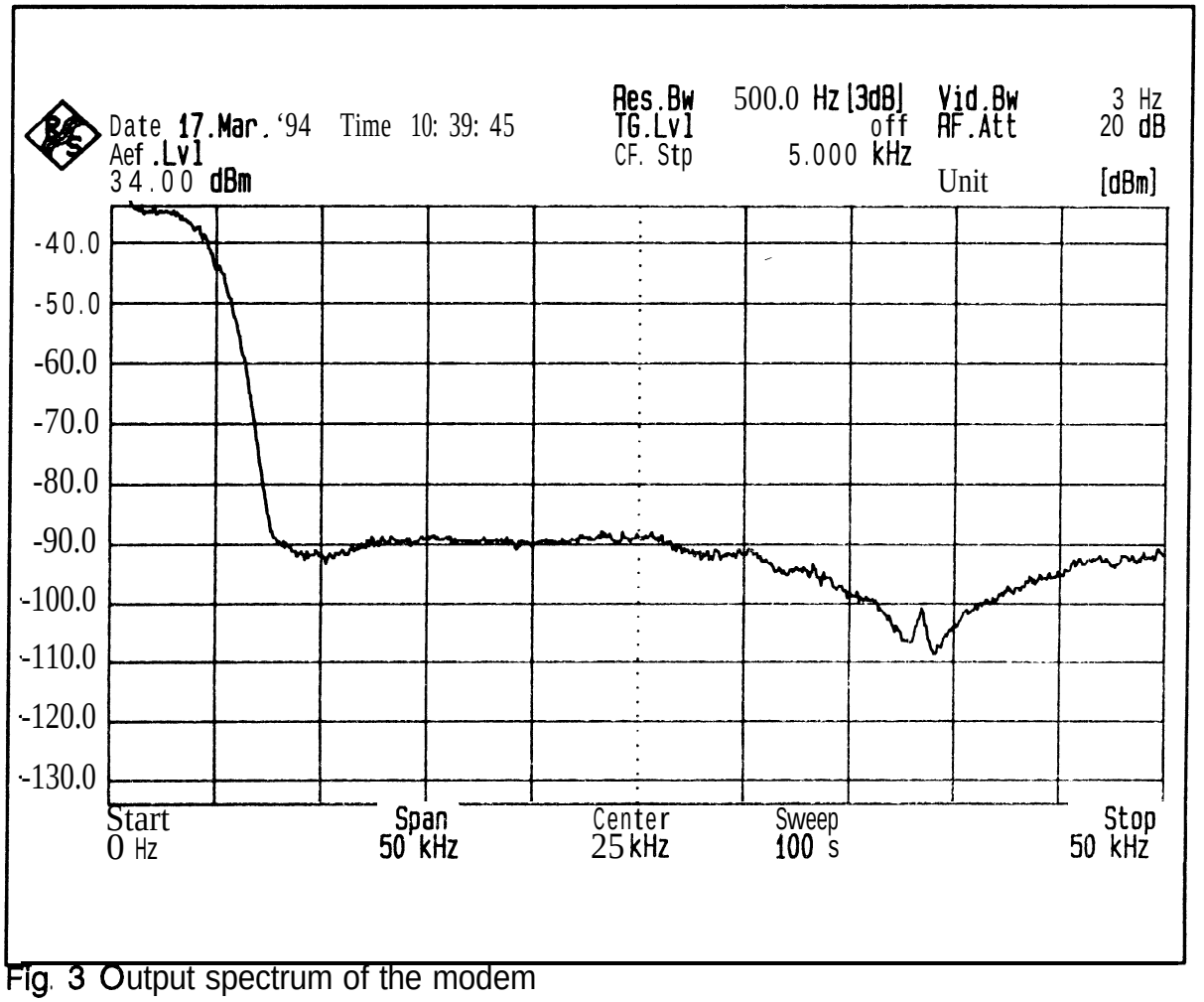


Fig. 3 Output spectrum of the modem

An implementation

We have implemented this modem on a European node controller board in 1993. The RMNC3 board contains the controller core with a HD63C09 processor, 64k RAM, 64k EPROM, an 85C30 for HDLC and a 65SC22 for a parallel bus to interconnect multiple controllers. In addition there was enough space for a TCM3105 for AFSK operation, a MAX232 for a KISS interface (e.g. BBS port) and the described scalable FSK modem.

It is a single port controller so that only one of the modems is used at the same time - with this configuration 99% of nowadays PR modulations can be realized with one board layout. Switching between the modems is not necessary because one proper board is used for each point-to-point link. Fig. 4 shows the asembled board (160x100mm, abt. 6.4x4 inches).

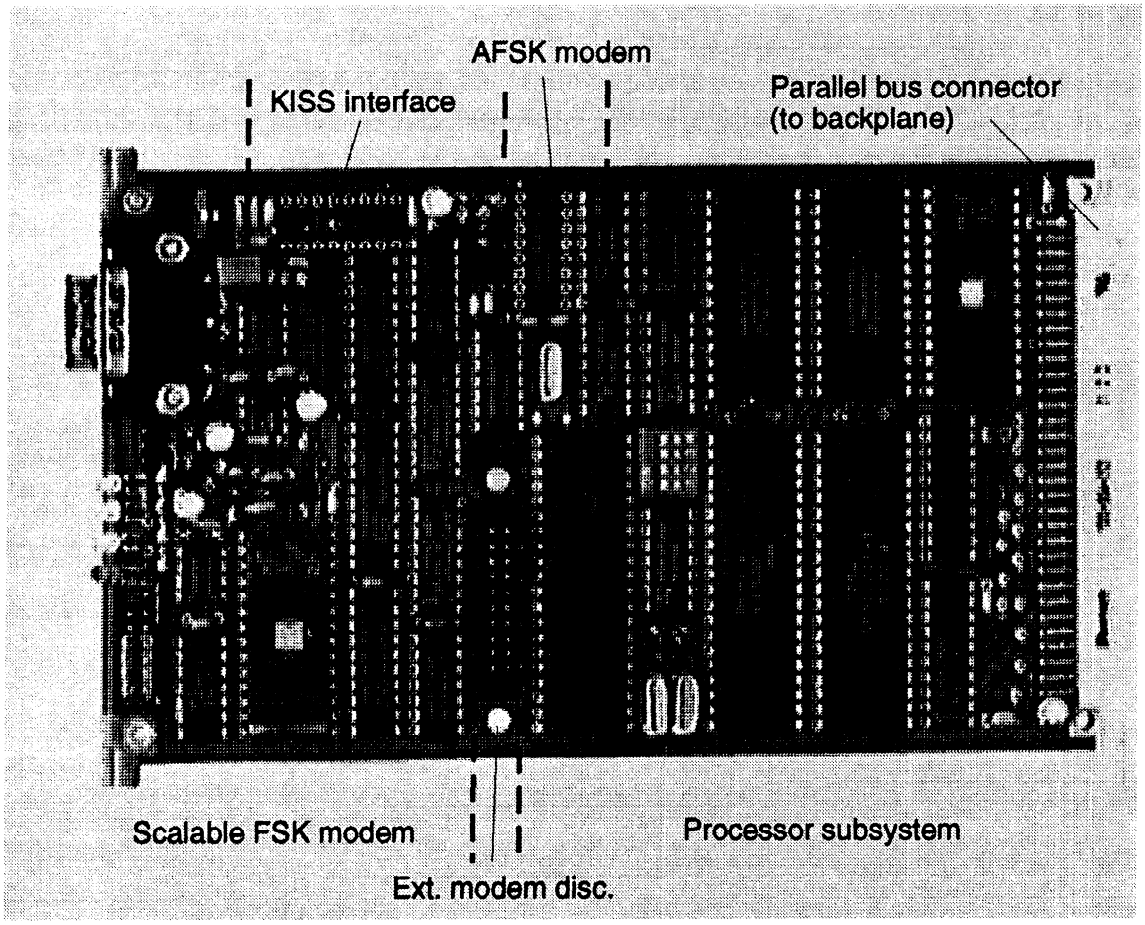


Fig. 4 Photo of an RMNC3 board with scalable FSK modem

Bibliography

[1] Rech, W.-H., DF9IC: Modernes FSK-Modem - kompatibel zum Standard nach G3RUH. ADACOM Magazin 211991, 13-31.

[2] Jost, G., DK7WJ: RMNC3: einer für Alle(s). ADACOM Magazin 6 (1993), 16-35.

[3] Rech, W.-H., DF9IC: Iterierte duale Filterung - ein Verfahren zur Synthese von Datensignalen. ADACOM Magazin 6 (1993), 44-47.

MacAPRS™

Mac Automatic Packet Reporting System
A Macintosh Version of APRS™

Keith Sproul, WU2Z
Mark Sproul, KB2ICI
Radio Amateur Telecommunications Society

Keith Sproul, WU2Z
698 Magnolia Road
North Brunswick, NJ 08902-2647
(908) 821-4828 Home
(908) 563-5389 Work
(908) 563-5035 Fax
Internet: akasbbl @ peabody.sct.ucarb.com
AppleLink: Sproul.K
Packet: WU2Z@KB4CYC.NJ.USA

ABSTRACT

MacAPRS is a Macintosh version of the popular *APRS*, Automatic Packet Reporting System, by Bob Bruninga, WB4APR, [1]. Bob introduced his APRS program at the ARRL CNC in 1992. APRS is a system that uses Packet Radio to track objects, using maps on computer screens. His version runs on Intel-based computers running DOS.

Since the introduction in 1992, this program has gained widespread popularity and has had many uses. The most obvious of these uses has been in public service events such as bike-a-thons and other public events covering large areas. It has also been used for such things as tracking amateur balloon launches and tracking the space shuttle.

This paper discusses both the improvements/enhancements to the APRS system and also the introduction of the Macintosh version. It also discusses many of the real-world applications of this system and future possibilities.

INTRODUCTION

Bob Bruninga, WB4APR, has developed a system for tracking objects using Packet Radio. His system uses unconnected packets (UI frames) for transmitting the position and other information about each station or object. He has been working on different aspects of this system since 1984. In 1992 Bob presented a paper at the ARRL Computer Networking Conference that introduced the PC program now called *APRS* (Automatic Packet Reporting System). *APRS* is a program that sends and receives these packets and displays the objects

on a map on your computer screen. Over the last two years, APRS has become quite popular on Packet Radio and is gaining new users daily.

Bob Bruninga wrote his APRS program in QuickBasic to run on Intel-based DOS computers. Since many Hams have Intel-based computers, this program started to gain popularity rapidly. However, Hams that had other computers could not participate.

Mark and I have been writing software for Ham radio for the Macintosh over the last several years We started with an experiment with a graphical user interface on the air in 1991,[2]. We proceeded with better ways to receive Packet Mail in 1992, [3], and in 1993, we presented two papers. One was on Packet Tracker, [4] a program that displays a real-time logical map of packet connections and the other was Mail Tracker, [5] which used maps to show where mail had been as it traveled across the country. We have also written map software back in the early 80's while in college. We wanted to get *APRS* running on the Macintosh so we decided to write the Macintosh version of this program.

MacAPRS is a Macintosh implementation of WB4APR's APRS protocols. *MacAPRS* is written in Think-C and was done as a completely separate program without using any of the code from Bob's original QuickBasic program. *MacAPRS* is designed to be fully compatible with the protocols developed by Bob Bruninga.

We have worked with Bob on formalizing the protocols used in these programs. We have agreed to keep our protocols in sync with each other and to not implement new variations without checking with each other and to try to release new features at or near the same time. We have also discussed several new features that we plan on implementing over the next year or so.

APRS SYSTEM OVERVIEW

The APRS programs, both Mac & PC, use Packet Radio to send and receive station position reports from a variety of different types of stations. The information contained in these reports vary with the type of station, but in general, they all have at least the Latitude, Longitude and station type of the sending station. These packets are sent as unconnected packets <UI Frames>. The APRS programs receive these packets and display an appropriate symbol on a map showing where the station is located.

In addition to being able to able to transmit your own station location, you can also send information about objects. For example, you can send out the position of a hurricane out in the ocean, giving its Latitude/Longitude and station type (Hurricane). Then everybody else listening will know where the hurricane is. This feature makes APRS very useful for many different applications.

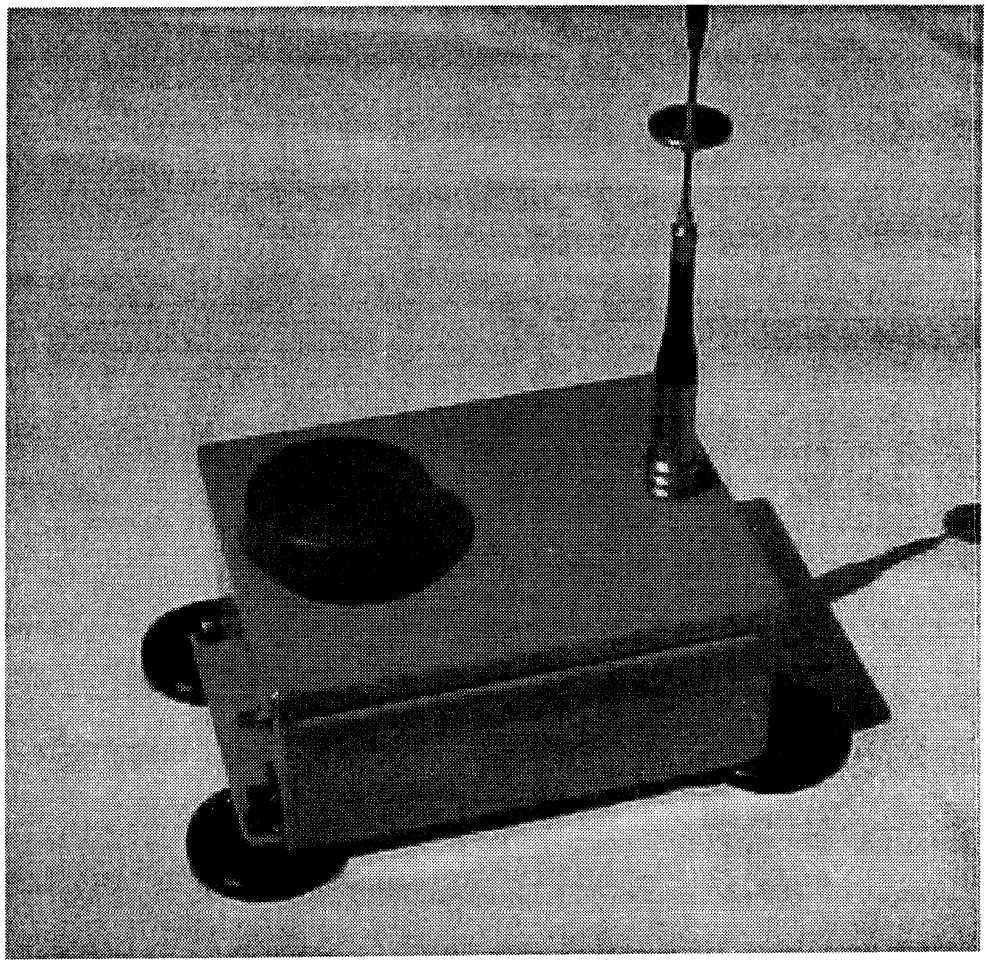
APRS has evolved over the last year or so and is being used in more and more applications. Many of the improvements have been enhancements of the user interface and the types of information that can be displayed. There have been many additions to the data and station types to handle the many different situations encountered in public events. There have also been lots of improvements in the quality and types of maps available. The early maps were created by hand, now we have the ability to use map data from the USGS (United States Geological Survey).

GPS and TNCs

PacComm has developed a new version of software in their TNC to support hooking up a TNC directly to a GPS unit. The commands are listed in the table below. This set of commands allows a TNC to send out a 'Beacon' containing the current position of the unit. As the unit moves, the GPS sends out updated position reports to the TNC. This data typically gets updated every second. Since we don't want the TNCs sending out the data that often, the PacComm commands make it so that the TNC sends this data out only as often as needed. Usually once a minute, or once every 5 minutes.

Using PacComm TNCs, we have made several self-contained units that have a GPS receiver, a TNC, and a radio. We installed the GPS, TNC, and Radio into a watertight box, and then put magnets on it so that it can easily be installed on a car. There are external connectors for power, antennas, and computer connections for both the TNC and GPS. These computer connections are for configuration only and are not used while the vehicle is in motion.

Figure 1: GPS - TNC - RADIO All-In-One Unit



PacComm GPS Related Commands: (Version 3.2 and later)

GPS	on/off	
GPSITEXT	<string to send to GPS on startup>	
GPSTEXT	$GPGGA	The default string to look for coming from the GPS
LOC	every x	How often to send out the GPS location
LPATH	call-string	Same as UNPROTO command, but for GPS data only
LTEXT		The string accepted from the GPS that will get sent.

To use this feature of PacComm TNCs you need to do the following:

- Set the UNPROTO path for the LOCATION with the LPATH command:
  LPATH APRS via **GATE,GATE,WIDE**
- Set the string to look for from the GPS unit (usually the default):
  GPSTEXT $GPGGA
- Set how often the location information gets sent out:
  LOC every 30
- Set GPS mode on:
  GPS ON

Now, every time the TNC gets turned on, it starts looking for the string entered with the GPSTEXT command. Each time it sees a line that starts with this text, it sticks that line into the LTEXT buffer. It then sends that text out via UI frames via the path specified in the LPATH command. It does this at the time interval specified by the LOC command.

If the GPS requires some special command to get it started, this can be set with the GPSITEXT command. If this is done, that command will be sent to the GPS every time the TNC is turned on. This makes this system very flexible.

Once all of this is done, you have a stand-alone system that will transmit its position using a GPS receiver, a TNC and a radio. No computer is required. See Figure 1 for one of the units we have built.

In addition to the commands above, you can also set the normal TNC Beacon Text and the normal Beacon rate etc. Be aware that when in GPS mode, the unit will NOT function as a Digi-peater. If you want to further identify what type of station the unit is, you can specify it's station type in the normal beacon text by putting the station-type character identifier in braces as the first three characters of the Beacon Text. i.e.:

- BTEXT {O} KA9NHL Balloon Launch

However, you must also set the beacon text rate and the UNPROTO path for this. The UNPROTO path SHOULD be the same as the LPATH, and the beacon rate should be much less often than the LOCation rate.

This set of commands makes PacComm TNCs very useful for GPS applications and allows them to be used with a wide variety of GPS units. If you already have a PacComm TNC, you can simply upgrade to Version 3.2 ROMS and get these features.

APRS for Public Events

APRS has been used in many public events. The most common places have been bike-a-thons, marathons, etc. Both the PC and Mac versions were used in the spring Multiple Sclerosis Bike-A-Thons this spring. We used the GPS-TNC boxes described above, putting one of these on the head vehicle and each of the tail vehicle(s). In the future, as we get more of the GPS-TNC units, we also plan on putting one on each of the emergency vehicles.

By having instant knowledge of where the people are, it is much easier to plan and be prepared for the many things that go on during these events. We have also found that when we have a lack of volunteers for 'shadows' to the race officials, it is quite often good enough to simply know where they, so if we are short handed, we can either assign a 'shadow' to someone, or put a GPS/TNC/Transmitter on their car. Later on, if that person is needed, we know exactly where they are.

APRS for Weather Monitoring

Many people, especially several groups in Florida, are using APRS for tracking weather. Many features have been added for this purpose. There are many station types specific to weather, such as Hurricane, Thunderstorm, Flood, Small Craft Advisory, Gale, etc. (See Table 2) In addition to being able to display simple weather information,' both the PC and Mac versions of APRS can be hooked up to a Peet Brothers Ultimeter II weather station. With this hookup, the programs will then send out the appropriate weather information automatically. The system can send out the windspeed and direction, rainfall, and temperature,. The wind direction and speed get displayed as a vector on the screen showing the relative speed and direction of the wind at that location.

APRS for Direction-finding

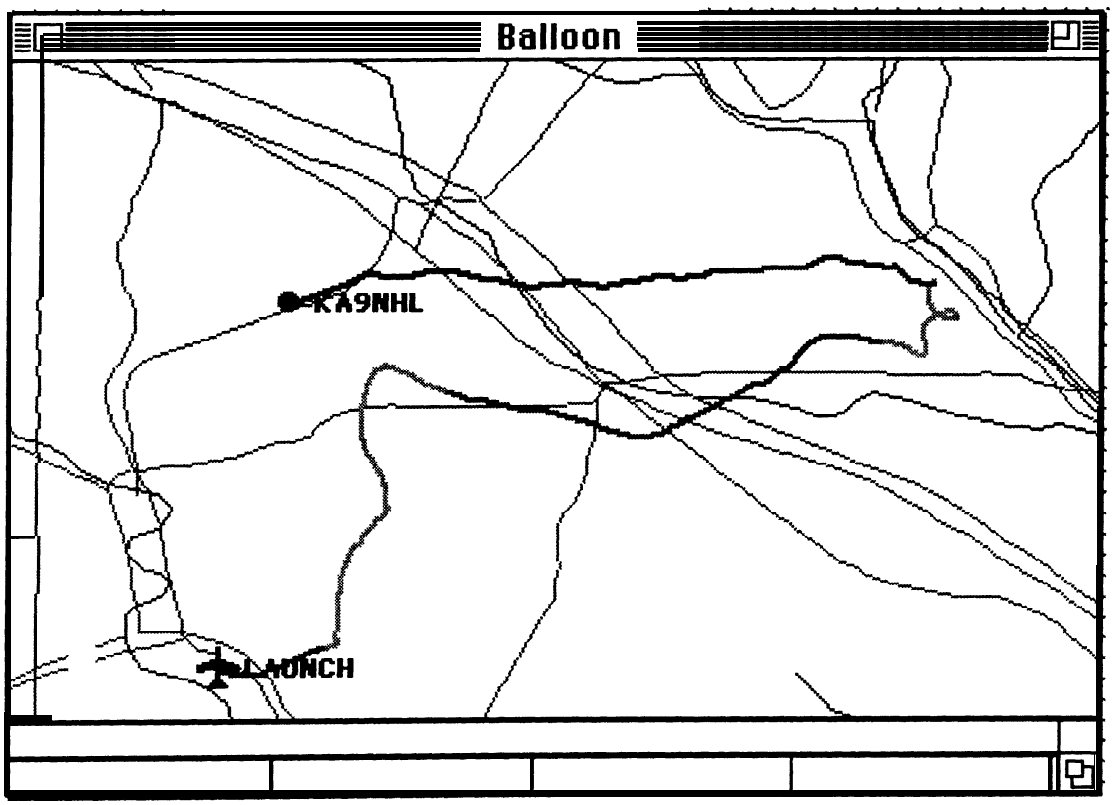
The APRS system can be used for Direction-Finding too. If a station has a direction-finding capability, or even a manual beam heading, it can send out that information as a bearing similarly to the wind information mentioned above. The programs will then display the station on the map with a line coming out at the appropriate angle. This line extends to the end of the map. If you have more than one station reporting this type of information, you can get an immediate graphical location of where the transmitting station is. There have been a couple interfaces developed to hook the Doppler direction-finding units directly into the computer to automate some of this procedure.

If you are having problems with jammers on your repeater, you could hook up two or three stations with the APRS system and Doppler antenna systems. Then every time someone transmits, your computer screen will automatically show what direction the signal is coming from, and if someone else has the same setup a reasonable distance away, they too will have a fix. Their APRS system will then transmit the position and bearing of the signal that they receive and you will now have a triangulated fix of the transmitting station. Since everything can be saved to log files, you now have graphical records of everything transmitting on the input of your repeater over an extended period of time.

APRS for Amateur Radio Balloon Launches

David Chesser, KA9NHL, was involved in a balloon launch this past June where they put a PacComm payload of a balloon. On the ground, they had computers receiving the APRS information, thus allowing full map tracking during the entire flight of the balloon. David used the Macintosh version of APRS for this project We added some additional support to *MacAPRS* just for balloons to allow for altitude tracking. The position and altitude information was quite impressive to see. See figures below. From the data transmitted by the GPS-TNC unit, they were able to determine the maximum altitude of the balloon, which was 86,000 feet. They also got full altitude tracking within the MacAPRS program. Although it is hard to tell in black and white, the balloon track in Figure 2 shows the relative height by the color of the line. The group that David is involved with plans on doing several more balloon launches this summer with GPSs in all or most of them.

Figure 2: KA9NHL Balloon Track



APRS MAPS

When Bob Bruninga first released his *APRS* program, all of the maps were done by hand. He had developed a simple and efficient method for creating map files. Since that time, there has been a high demand for better and better quality maps. All of *MacAPRS* maps were created from data from the USGS (United States Geological Survey). The PC version now also has the ability to create maps using this USGS data. This has allowed for the creation of much better maps. We are continuing to improve the types of maps used and look forward to significant improvements over the next year.

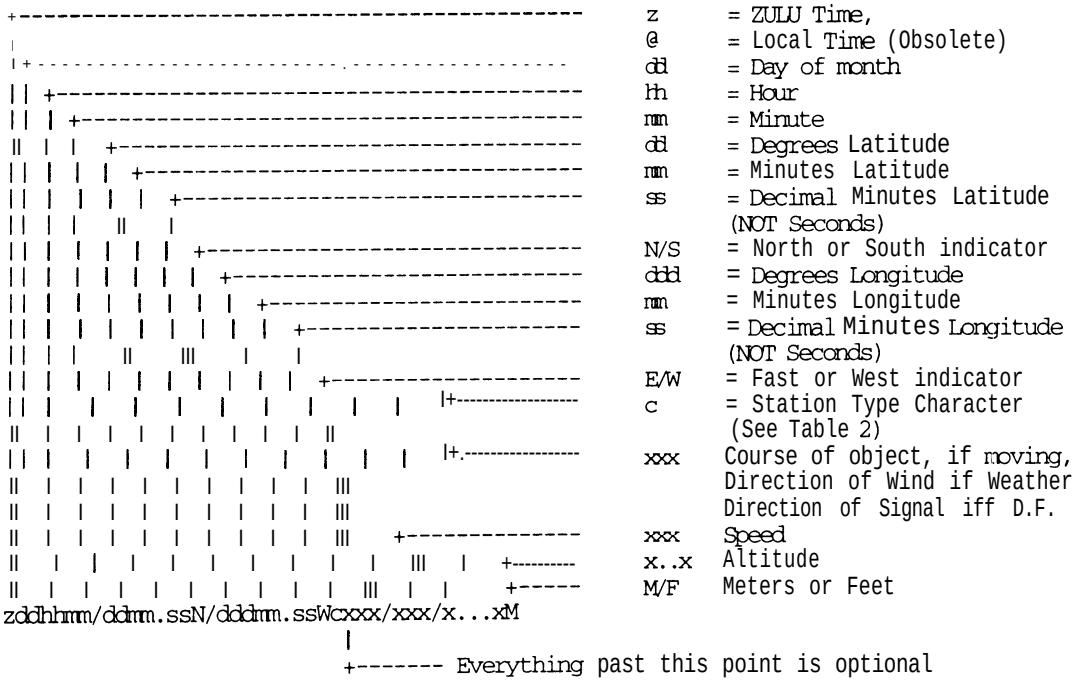
In addition to the USGS Map files, *MacAPRS* can import all of the map files used in the PC version. This allows for the exchange of map files between the two systems.

APRS PROTOCOLS

The APRS programs transmit data using AX.25 UI frames, i.e. Unconnected Packets. The information sent in these packets includes many different aspects of what is needed to track stations. The amount and types of data vary widely with the types of stations being tracked. The following discussions explain the basic information contained in these packets and some of the many different variations.

Most of the packets used in APRS include the position information,, This is explained in the table below. In addition to the position information, there is a STATION TYPE identifier. The current list of recognized station types is given in Table 2.

Table 1: Position Protocol Definition



The following are some examples:

```
WU2Z*>GATE>GATE>WIDE>APRSM:z272146/4026.98N/07428.70WM MacAPRS
VE2JOR>GATE*>WIDE>APRS:@271912/4535.04N/07333.33W_090/000/T065/U-II (auto)
N4PUQ*>GATE>WIDE>APRS:@270602/3602.04N/08418.69W-Oliver Springs,Tn.
```

Table 2: Station Type Character Definitions

Dec	Hex	Char	Description	Dec	Hex	Char	Description
33	21	!	Emergency	80	50	P	Police Car
34	22	"	Rain	81	51	Q	Earthquake
35	23	#	Digipeater	82	52	R	RV - Recreational Vehicle
36	24	$	Sunny	83	53	S	Space
37	25	%	DX Cluster	84	54	T	Thunderstorm
38	26	&	HF/VHF Gateway	85	55	U	Bus
39	27	'	Air Plane	86	56	V	Unused
40	28	(	Cloudy	87	57	W	Unused
41	29	)	Hump	88	58	X	Helicopter
42	2a	*	Snowflake	89	59	Y	Yacht
43	2b	+	Red Cross	90	5a	Z	Unused
44	2c	,	Jay	91	5b	[	BBS
45	2d		QTH/House	92	5c	\	Direction Finding
46	2e	.	Small Dot	93	5d	]	Mailbox, PBBS, etc
47	2f	/	Default Symbol	94	5e	^	Unused
48	30	0	Black Square	95	5f		Weather Report
49	31	1	Brown Square	96	60	`	Thunderstorm
50	32	2	Red Square	97	61	a	Ambulance
51	33	3	Orange Square	98	62	b	Bicycle
52	34	4	Yellow Square	99	63	c	Unused
53	35	5	Green Square	100	64	d	Fire Department
54	36	6	Blue Square	101	65	e	Sleet
55	37	7	Violet Square	102	66	f	Fire Truck
56	38	8	Gray Square	103	67	g	Gale Warning
57	39	9	White Square	104	68	h	Hospital
58	3a	:	Fire	105	69	i	Unused
59	3b	;	Portable (Tent)	106	6a	j	Jeep
60	3c	<	Small Craft Advisory	107	6b	k	Truck
61	3d		Trains	108	6c	l	Unused
62	3e	>	Vehicle	109	6d	m	Color Apple Logo (Reserved)
63	3f	?	Unused	110	6e	n	Unused
64	40	@	Hurricane	111	6f	o	Circle/Default
65	41	A	Brown Circle	112	70	p	Partly Cloudy/Partly Sunny
66	42	B	Red Circle	113	71	q	Unused
67	43	C	Orange Circle	114	72	r	Radio Antenna
68	44	D	Yellow Circle	115	73	s	Ship
69	45	E	Green Circle	116	74	t	Tornado
70	46	F	Blue Circle	117	75	u	U shape (Submarine)
71	47	G	Violet Circle	118	76	v	Van
72	48	H	Gray Circle	119	77	w	Flooding
73	49	I	White Circle	120	78	x	Nodes
74	4a	J	Black Circle	121	79	y	Unused
75	4b	K	School	122	7a	z	Three Horizontal Bars
76	4c	L	Unused	123	7b	{	Fog
77	4d	M	Color Apple Logo	124	7c	|	Don't Use (TNC Switch Char)
78	4e	N	Unused	125	7d	}	TCP/IP
79	4f	O	Balloon	126	7e	~	Don't Use (TNC Switch Char)

This list of station types has evolved over the last couple of years. Many of the items have been added to facilitate public events and emergencies. Prime examples of this are the bicycle that was added for MS bike-a-thons and the Fire that was added for the California Fires The balloon was added because Hams doing balloon launches have started to put APRS transmitters into their balloon systems. The series of colored circles and squares were added for public events where you need to know which station was which, but they were so close together that the call signs would overlap and become unreadable.

Macintosh version of APRS, MacAPRS

The Macintosh version of APRS was released at Dayton Hamvention this past April. This version has all of the functionality of the PC version of APRS and has many new features specific to the Mac and its environment. The Macintosh version is fully compatible with the PC version of APRS and has many enhancements not possible in a non-windowing environment.

The first and most obvious difference is the capability to have multiple map windows open at the same time. The advantages of this are many. The most common configuration is to have a wide area map and one or two local area maps open at the same time. (See Figure 3) You also have full mouse support, which is a requirement in any Mac application, and you have the ability to simply click on a station to get basic information about that station, or to double-click on it to get full information on it. Double-clicking on a station icon brings up an Information-Window with all of the statistics known about that station. In addition to the APRS information about a station, if the user has the Buckmaster CD-ROM, the program will do an automatic lookup of the call sign and display the full name and address of the person selected. This feature has proved extremely useful. (See Figure 5) You can also bring up several other windows such as a Station List, (Figure 4), and Message List, (Figure 6).

The Macintosh version requires System 7.0 or later and wants 4meg of memory, however, it will run in less. It will not run on a Map Plus or Mac SE. Like most Macintosh applications, it also supports multiple monitors, this allows you to have several large maps on different computer screens at the same time. This can be quite useful for large spread out events. You can have as many different map windows open at the same time as you have memory.

Like the PC version of APRS, MacAPRS has full GPS support and support for the Peet Brothers Ultimeter-II. With GPS support, you can hook up a GPS unit to your computer and track where you are going while in a vehicle. This works out very well with Macintosh PowerBooks, the portable/notebook Macintosh line of computers.

In addition to the normal Lat/Lon input of the GPS, MacAPRS also supports some statistics and other data that is available from some of the GPS units on the market. This is all done via the NMEA 0183 specification, (NMEA, National Marine Electronics Association). In conjunction with the GPS input, you can automatically set the computer time from the satellites which will give very hi accuracy time once you have locked into a GPS satellite.

Figure 3: Multiple Map Windows

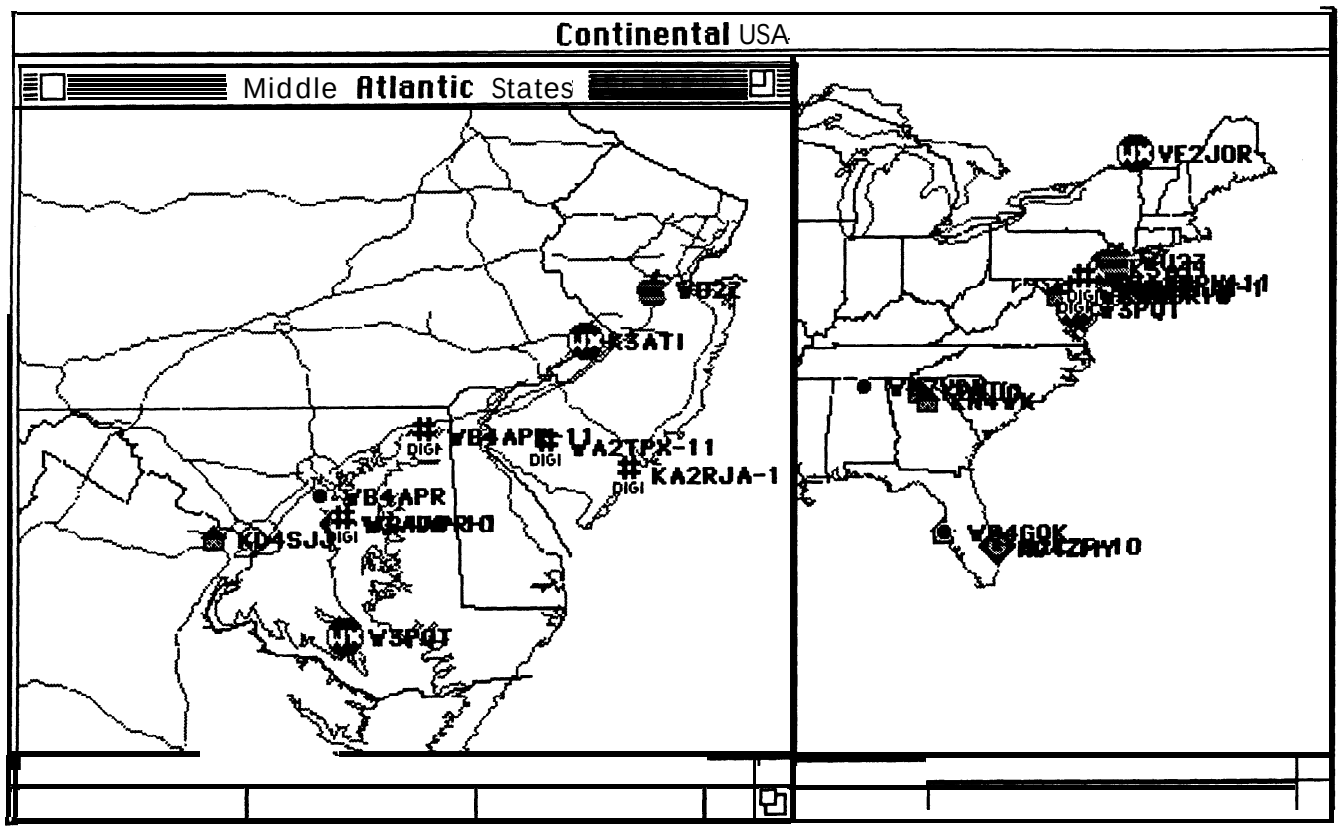


Figure 4: Station List Window

Station List

	Call	C	A	T	O	Count	ID String
	WU2Z	M				257	Keith, North Brunswick, NJ
	N6OAA	;				40	
	VE3TQX					121	
	KN4WK	-				19	KN4WK/R ECHO/D KN4WK-1/B
	N2CZF-10	&				113	ECHO, N2CZF-10 digipeater, GP
	N9GBJ	-				14	
OBJ	N2VEP-4	>			=	0	Station Added Manually
DIGI	KA2RJA-1	#				17	
	K3ATI	-				28	
	N2IPH					21	N2IPH/R
DIGI	WB4APR-11	#				5	
DIGI	WA2JNF-2	#				1	
	N3HTZ	-				4	
DIGI	WB4APR-1	#				6	

Figure 5: Station Information Window with CD-ROM lookup

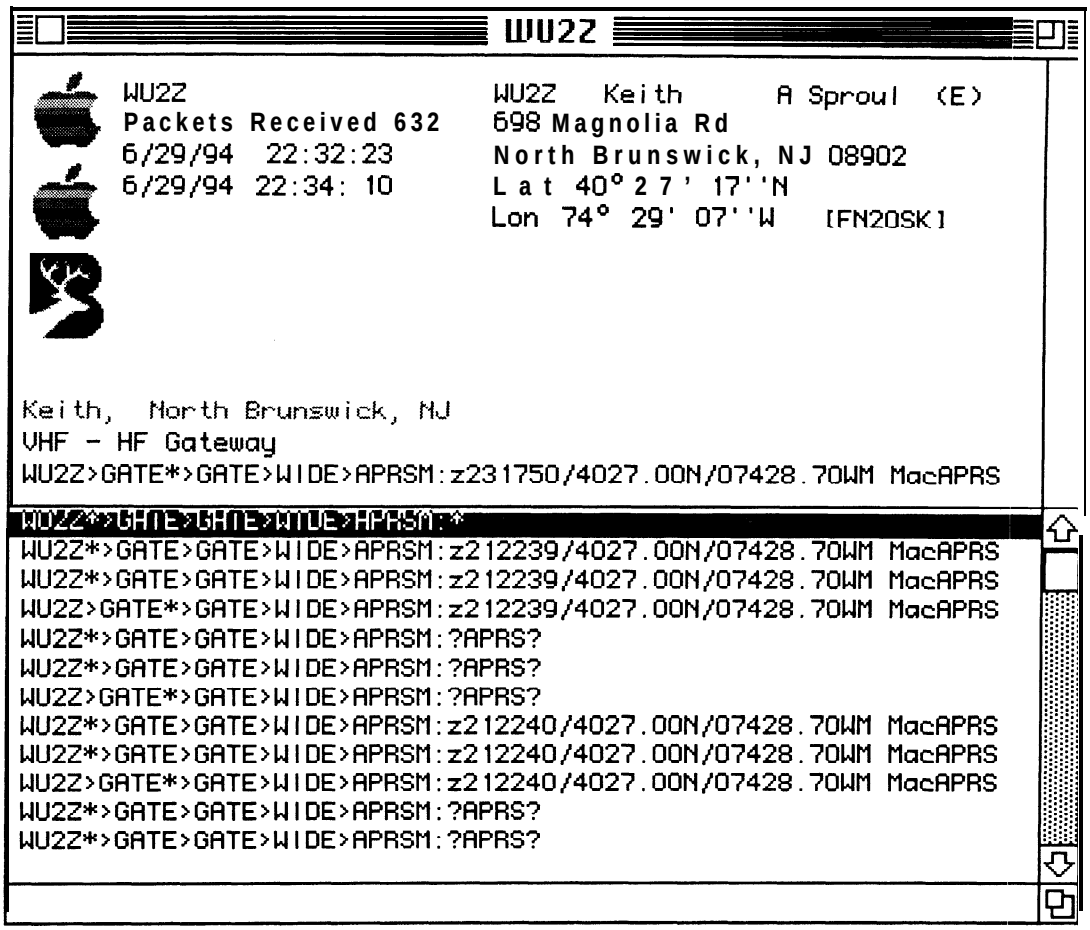
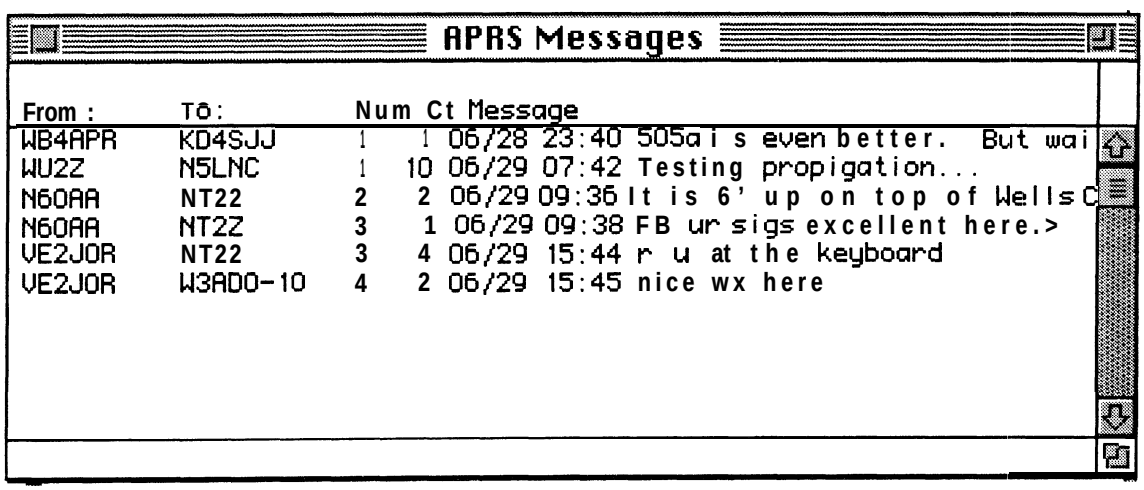


Figure 6: Message List



MacAPRS Databases

In addition the information received over the air, there are other databases built into the program as well. You can locate several different things that are of interest to Hams. If you have the Buckmaster CD-ROM, you can do a FIND and type in a call sign, it will show you on the map where that person lives and tell you their name. You can also locate any zip code in the US, and any airport, simply by typing in the appropriate item you want to find. All of these searches show the appropriate location on the map. The Grid Square option shows either 2-letter Grids or 4-letter Grids. (See Figure 7).

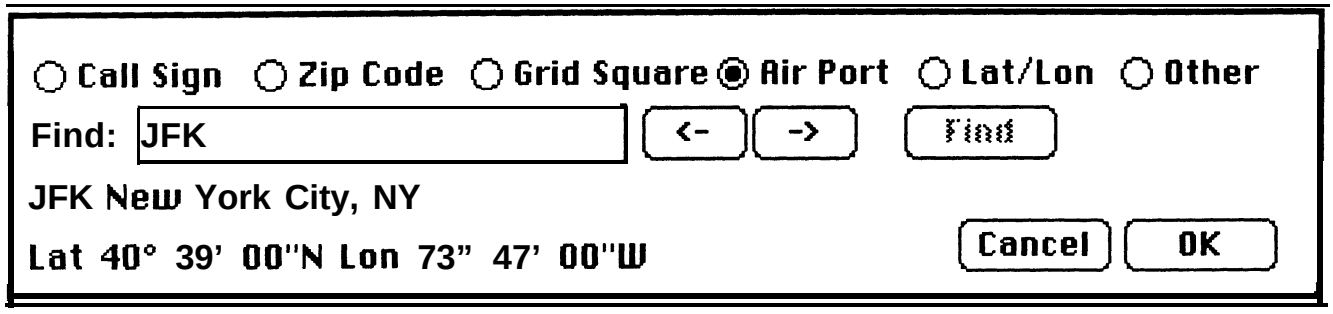
As you type in what you are looking for, the program automatically figures out which type of information you are entering and selects the appropriate category. The categories supported in the FIND function are:

- Call Sign (If you have the Buckmaster CD-ROM)
- Zip Code
- Grid Squares, both 2-letter and 4-letter
- Airports (Search by 3-letter or 4-letter Airport Code)
- Latitude/longitude

For Zip Code and Airport data, it will automatically fill in the Lat/Lon as soon as it recognizes what you have typed. Once it recognizes the data as an Airport or Zip Code, you can then push the arrow keys to scroll through to the next or previous airport or zip code.. After you have what you want, hitting okay will show were it is on the map. The airport database includes over 3000 foreign airports and 18,000 US airports. It also has altitude for the US airports. The zip code database has all of the valid Zip Codes in the US.

For call sign, you type in a call, you then can push the FIND button and it will do the look up. After you have what you want, you hit OK and it will show you that location on the map. If you just hit OK, it will simply go look it up and show you the location. The find button actually brings up the name etc. within the dialog box before it shows you the location on the map.

Figure 7: Find Airport



APRS FUTURES

I am trying to get better map data for other parts of the world. This project is already in progress, and proceeding nicely. I would also like to get a database of the postal codes of Canada with latitude and longitude, but so far, have not been able to locate this data. I have the US zip code database, and would like to be able to extend this capability to Canada.

One thing that the APRS community needs is for someone to come out with an inexpensive, data only (SSB) 30 meter radio. There are several companies that have nice radios that are CW only, but we need a small, crystal-controlled radio that we can run on 30 meters for transmitting APRS data from the car on HF. This radio does not have to be fancy, just small, inexpensive, and able to output at least 25 watts.

We are working with Peet Brothers to further enhance the weather capabilities of the system and we are working with the people making the interfaces to the Doppler antennas to enhance that capability also.

REFERENCES

[1]
Automatic AX.25 Position and Status Reporting, Bob Bruninga WB4APR, ARRL. 11 th, 1992 Computer Networking Conference, November, p 13-18, Teaneck, New Jersey.

[2]
GUI Packet, Graphical User Interface On Packet Radio, Keith Sproul, WU2Z, and Mark Sproul, KB2ICI, ARRL 10th, 1991 Computer Networking Conference, p 137-147, San Jose, California.

[3]
Sophisticated Mail User Interface Systems, Keith Sproul, WU2Z, and Mark Sproul, KB2ICI, ARRL 11 th, 1992 Computer Networking Conference, p 89-97, Teaneck, New Jersey.

[4]
Packet Tracker, A Graphical Packet Tracking Program, Mark Sproul, KB2ICI, ARRL 12th, 1993, Digital Communications Conference, p 77-82, Tampa, Florida.

[5]
Mail Tracker, A Graphical Mail Tracking Program, Keith Sproul, WU2Z, ARRL 12th, 1993. Digital Communications Conference, p 83-96, Tampa, Florida.

Formation of the TAPR Bulletin Board System Special Interest Group

David A Wolf, WO5H

ABSTRACT

Recognizing that future improvements to BBS operation were being hindered by a lack of central resource which people could consult for help and the exchange of ideas, the Board of Directors of Tucson Amateur Packet Radio, Incorporated[1] created a special interest group to study methods to address this issue. This paper describes the formation of the TAPR BBS SIG, its purpose and its progress to date.

There has been a growing awareness that the BBS forwarding network is not as efficient as it could be. Excellent ideas for improvement arise sporadically from individuals, regional groups and software writers, but they often don't get the exposure they deserve. Sentiment has been expressed that BBS sysops would benefit from a nationally-sponsored group to focus on its interests and to serve as a readily-accessible resource of information and education.

On March 4, 1994, the Board of Directors of Tucson Amateur Packet Radio Corporation, Incorporated, a non-profit research and development organization, voted to form a special interest group (BBS SIG) to become more directly involved in the issues concerning the operation of Bulletin Board Systems. The Board recognized that the operational experience of many of TAPR's members was a valuable and underutilized resource. The introduction of future hardware and software developments would be enhanced by TAPR's sponsorship of such operational support.[2]

The TAPR BBS SIG met for the first time on March 5, 1994, in Tucson, Arizona, in conjunction with TAPR's annual meeting held that weekend. 20 people attended this session. The primary purposes of the meeting were to determine if (1) there was sufficient interest among sysops to get together and (2) determine if a group of sysops *could* sit down and rationally discuss those things that affected the operation of their packet bulletin board systems. Happily, 20 people proved this possible. The basic direction and purpose for the TAPR BBS SIG was established during this first meeting.3

The TAPR BBS SIG is intended to provide a focus for the discussion of issues and formation of policies which, hopefully, will result in more efficient message forwarding between bulletin boards. Among the significant activities and goals of the SIG are:

(1) to collect and archive information from its participants (such as software bug reports), so that individual sysops don't have to each 'reinvent the wheel' every time they install or upgrade BBS software,

(2) provide an on-going on-line forum (on the Internet) for BBS sysops and others interested in BBS operation, [4]

(3) conduct workshops for BBS sysops at national gatherings, wherever practical, for the purpose of information gathering and exchange,

(4) make recommendations to amateur digital enthusiasts, the ARRL Digital Committee, and BBS software developers to facilitate more efficient forwarding between bulletin board systems,

(5) encourage consideration of how existing packet bulletin board systems might be integrated more closely (and more efficiently) *into* the evolving digital communications network, rather than continue primarily as *users* of the network,

(6) though the matters of message content and assignment of responsibility are of concern to most sysops, the TAPR BBS SIG is most likely to be successful if focused on operational matters,

(7) develop surveys and poll statistically-significant numbers of BBS sysops, as necessary, to build a profile of the typical BBS station, determine popularity of software, message flow rates, etc.

Some History and Background

The entire packet forwarding network basically 'happened.' There wasn't a grand design upon which to model the development of the network or the software. The first WORLI bulletin board system went on the air just over 11 years ago. Today there are a half-dozen major BBS programs in use throughout the world.

For the most part, BBS software writers and sysops have concentrated on getting individual systems to work. If messages resided only on the bulletin board they originated on, we'd never have to worry about matters such as addressing, 'to' and 'at' fields, BIDs, and such. Sysops tend not to think like distribution managers.

Some message identification and forwarding methods have been borrowed from other data services (such as the Internet, landline bulletin boards, and online services). Others have been developed to meet the unique requirements of amateur packet radio. Many good ideas have been rejected due to regional prejudice or the 'not invented here' syndrome. As BBS forwarding matures, there will be less opportunity to experiment to determine if some alternative addressing method might work better than those in popular use.

Traditionally, hams are very independent and. have difficulty adopting internally-created standards. The biggest contribution that the TAPR BBS SIG can make to amateur radio will be to resolve debates over relatively trivial matters such as "should it be NA or NOAM?" Rather than judging the adoption of conventions as threatening to individual independence, it is hoped that most sysops will be glad to be rid of the uncertainty. Sysops will then be able to concentrate on more productive activities, such as user education, improving their stations, better educating themselves, and enjoying BBS operation as opposed to burning out.

Accomplishments to Date

A quick review of the first meeting on March 5, 1994, in Tucson, is reported above.

Several Internet mail lists were established by TAPR to help facilitate communications between the various TAPR special interest groups and working groups. These lists currently include bbssig, netsig, and tapr-bb. They are open for general subscription.

Over 40 people participated in the TAPR BBS SIG meeting at the Dayton Hamvention on April 29, 1994. These resolutions were adopted at this meeting:

(1) the group reaffirmed the use of two-letter continental designators by stations currently using them until the matter could be studied in greater depth,

(2) the TAPR BBS SIG should study and issue a list of the most-frequently-used 'to' and 'at' fields,

(3) the TAPR BBS SIG should recommend a common flood bulletin structure.

A smaller working group[5] met the next evening to process the previous evening's meeting and determine how best to implement the resolutions. The concept of a set of recommendations in the form of a bbs setup and operating guide for the popular bulletin board software programs was developed.

An invitation for several volunteers to step forward to collect data, write sections of the Sysops' Guide, and moderate the TAPR BBS SIG Internet forum was posted on the Internet and the packet network just prior to the completion of this paper in late June.

Conclusion

TAPR's Board of Directors is to be credited for its willingness to provide leadership in the organizational aspect of digital communications.

Though our hobby is a communications-oriented one, amateurs often are poor communicators. The TAPR BBS SIG gives a wide cross-section of amateurs involved in digital communications an opportunity to exchange facts and opinions on line, on the air, and in person.

To be avoided is to quickly unleash a set of 'standards' on the packet community for immediate implementation The preparation of guidelines should be an open process with reasonable time for consideration and modification.

Because several different sets of 'standards' have been adopted on a regional basis, there is a demonstrated willingness by large groups of sysops and software developers to compromise for the benefit of the whole. This precedent is encouraging in that it indicates it is possible for the TAPR BBS SIG to be able to build a national consensus among key players.

References

18987-309 East Tanque Verde Road, Tucson, Arizona 85749-9399.

2,3 *see Packet Status Register,* Issue #54

4 to join the bbssig mail list, send an e-mail message to 'listserv @tcet.unt.edu' and include in the body of the message, the text:

join bbssig

5 Jay Ferron, N4GAA; Shelton McAnelly, KD5SL; Adam Tate, AB5PO; David Wolf, WO5H

Acknowledgments

Thanks to all who have participated in the live BBS SIG meetings, on-line forums and provided answers to questions. Many thanks, too, to a few pioneers who are willing to answer questions for us late arrivals to the packet party.

How Amateur Radio Operators Can Emulate an HF ALE Radio

David R. Wortendyke, N0WGC
U.S. Department of Commerce
NTIA/ITS.N1
325 Broadway
Boulder, Colorado 80303

Automatic Link Establishment (ALE) techniques became popular for Government HF radios four years ago with the adoption of Federal Standard 1045, and a Government effort to test all candidate HF ALE radios for interoperability and performance before major procurement actions could occur. Testing ALE radios in 1990 was a cumbersome process. As a spin-off the Institute for Telecommunication Sciences (ITS) in Boulder, Colorado investigated new techniques to simplify the method of testing. Last year ITS produced an audio compact disc (CD) with over 50 tracks of computer generated ALE modem tones. These ALE tones exercise almost all of the primary functions and addressing combinations required by the HF ALE Radio Federal Standard. All that is required to simulate an ALE radio is the CD, a CD player, and an HF transceiver with a microphone VOX input. As part of the development effort to produce the ALE modem tones, a computer program was written which makes PC "wave" files, or the digital audio (DA) files for the CD. The wave files may be played using a good quality PC sound card. This software uses ASCII text protocol files to produce the sound files, and the complete software package is now available free of charge on the Internet as of July 1, 1994. This paper describes the software use, and some of the future potential, for emulating an ALE radio using only typical amateur equipment.
Additional information via e-mail: ale-cd@its.bldrdoc.gov.

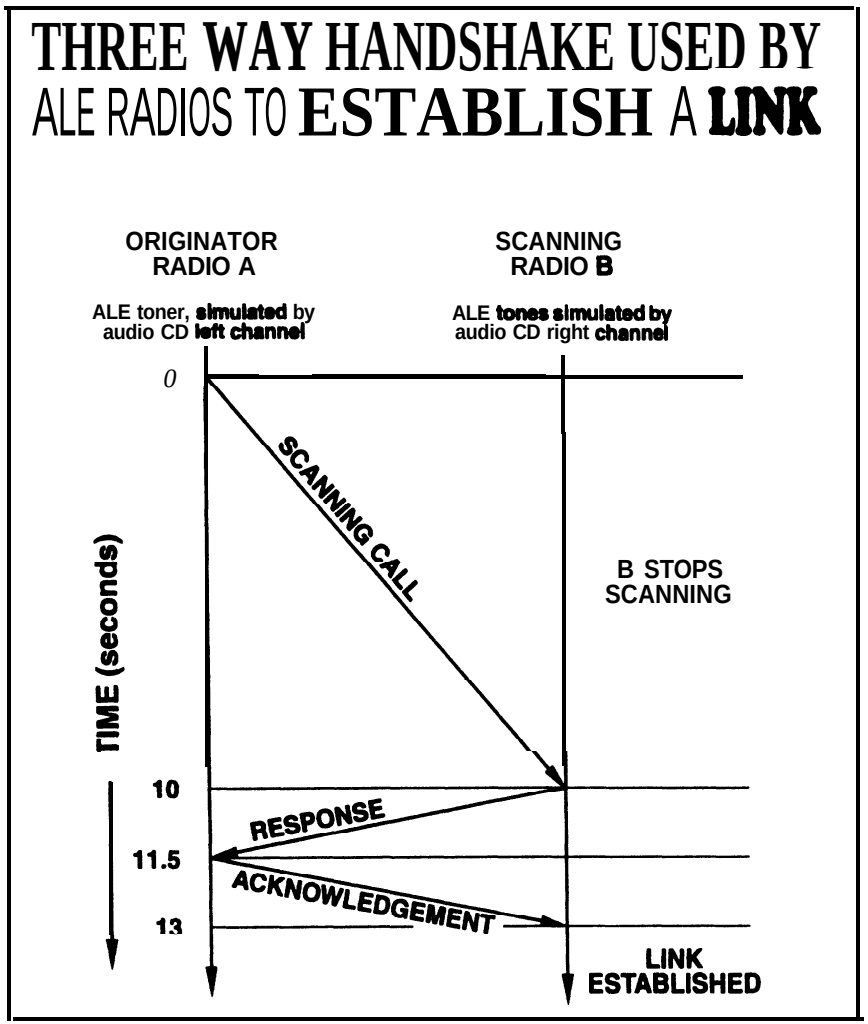


Figure 1

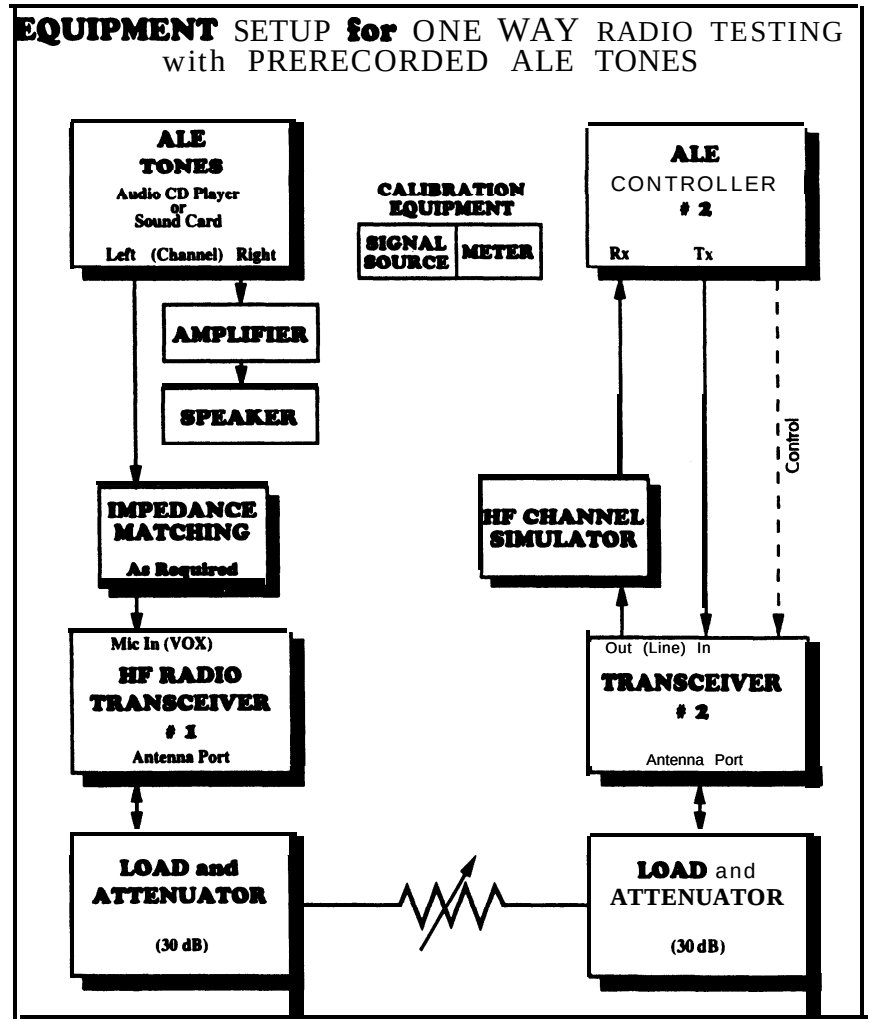


Figure 2

How Amateur Radio Operators Can Emulate an HF ALE Radio
Presented by
David R. Wortendyke, N0WGC

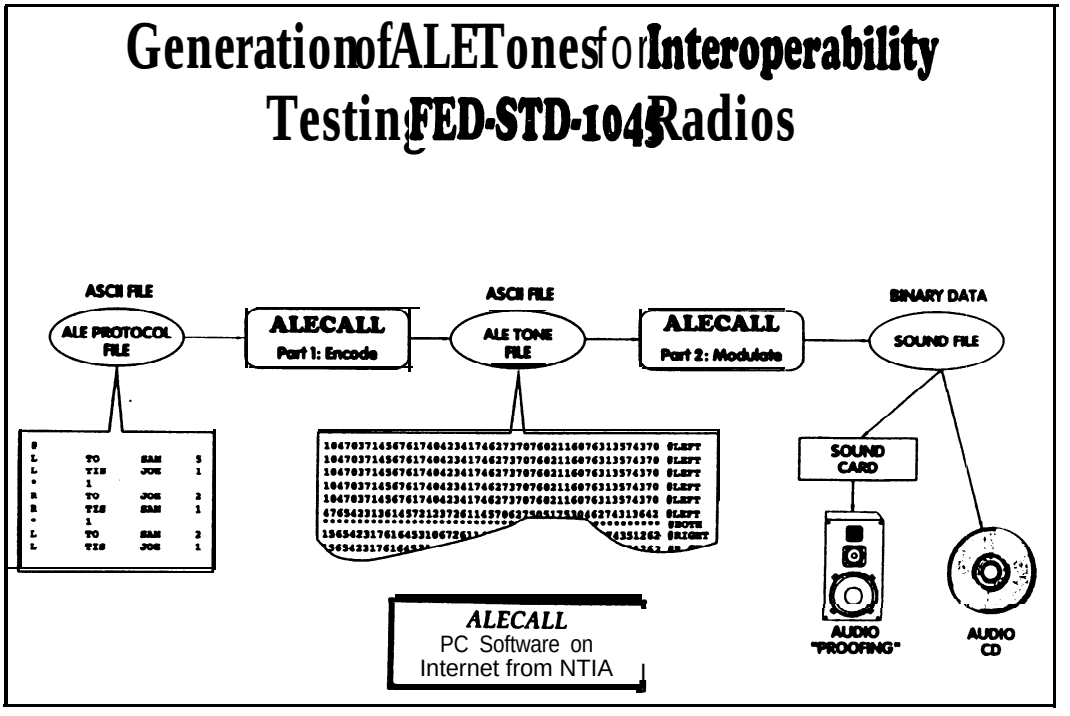


Figure 3

How Amateur Radio Operators Can Emulate an HF ALE Radio
Presented by
David R. Wortendyke, N0WGC

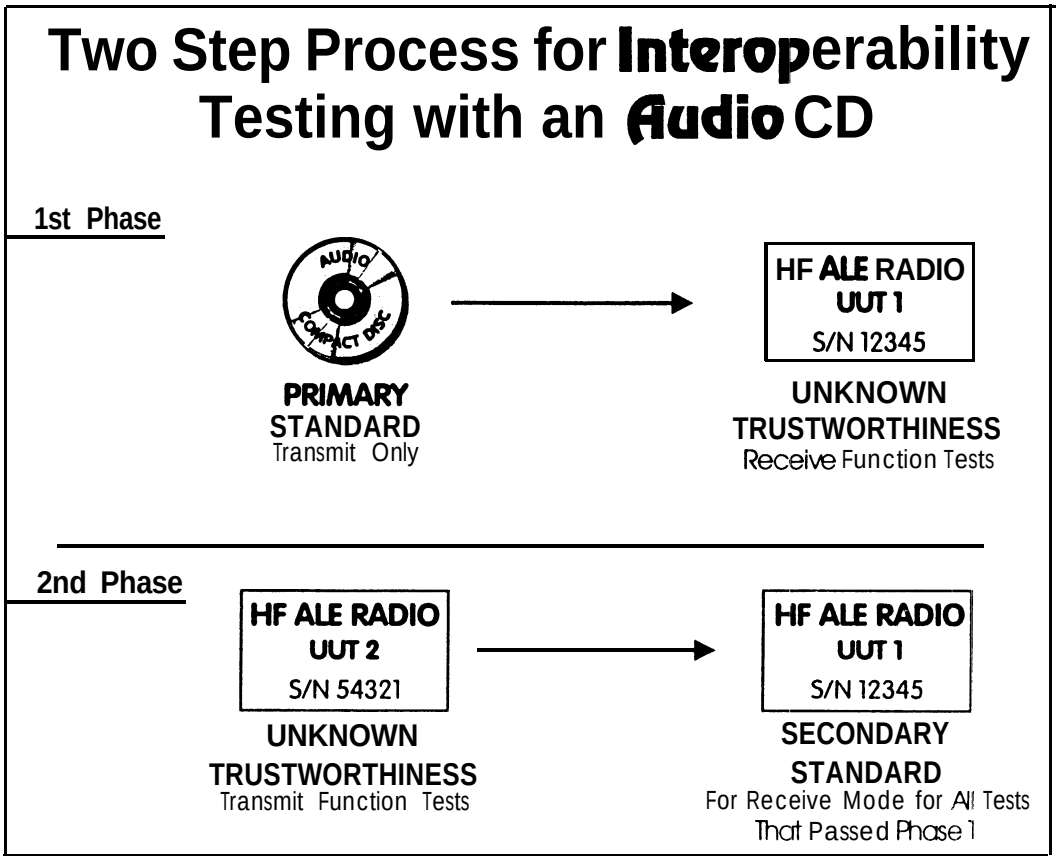


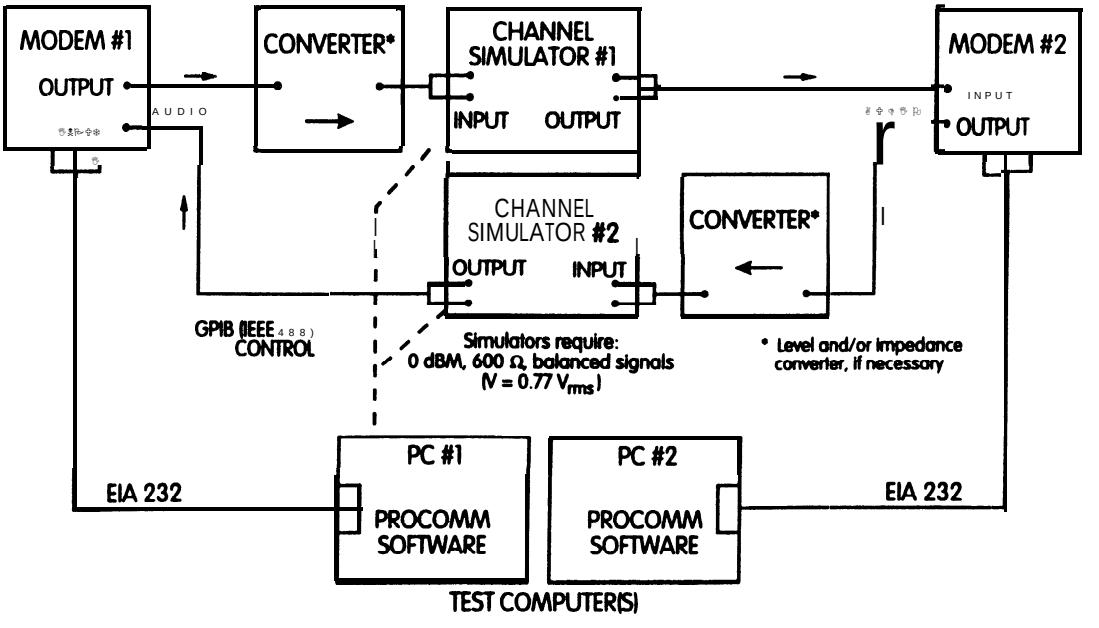
Figure 4

A Preview of HF Packet Radio Modem Protocol Performance

Teresa Young
Stephen Rieman
David Wortendyke, NOWGC
US Dept. of Commerce, NTIA/ITS
325 Broadway
Boulder, CO 80303

Many tests have been conducted over-the-air using various modem protocols designed specifically for HF radio links. It is impractical to compare the over-the-air performance of the protocols, primarily because the atmospheric propagation path conditions are always dynamically changing (non-stationary statistics). Engineers at the Institute for Telecommunication Sciences (II'S) developed a Windows program on a desktop PC to conduct controlled laboratory testing of modem protocols. Using this automated test program, we subjected the modems to a repeatable set of simulated propagation paths for a wide range of signal-to-noise (S/N) ratios. The six protocols tested were: AX.25, AMTOR, PACTOR, SITOR, CLOVER II, and Baudot. The ionospheric propagation conditions were simulated by two narrow-band, Watterson model, HF propagation channel simulators. Clear channel paths through the simulators and three degraded conditions were used: Gaussian noise, CCIR Good paths, and CCIR Poor paths. Over 3000 data file transfers were performed in a randomized manner at various S/N ratios for each of the six protocols. Both ARQ and broadcast mode were used when appropriate. All files received with an error were preserved so an extended computer analysis could be performed. Two metrics were chosen to evaluate the performance of the protocols: 1. throughput, a measure of the data transfer rate, and 2. errors, an indicator of the effectiveness of the protocol. The two metric parameters are compared for each protocol, various channel conditions, and signal strengths. A short preview of the data is provided by this paper.

Performance Testing of HF Modems
Equipment Setup



DATA FILES

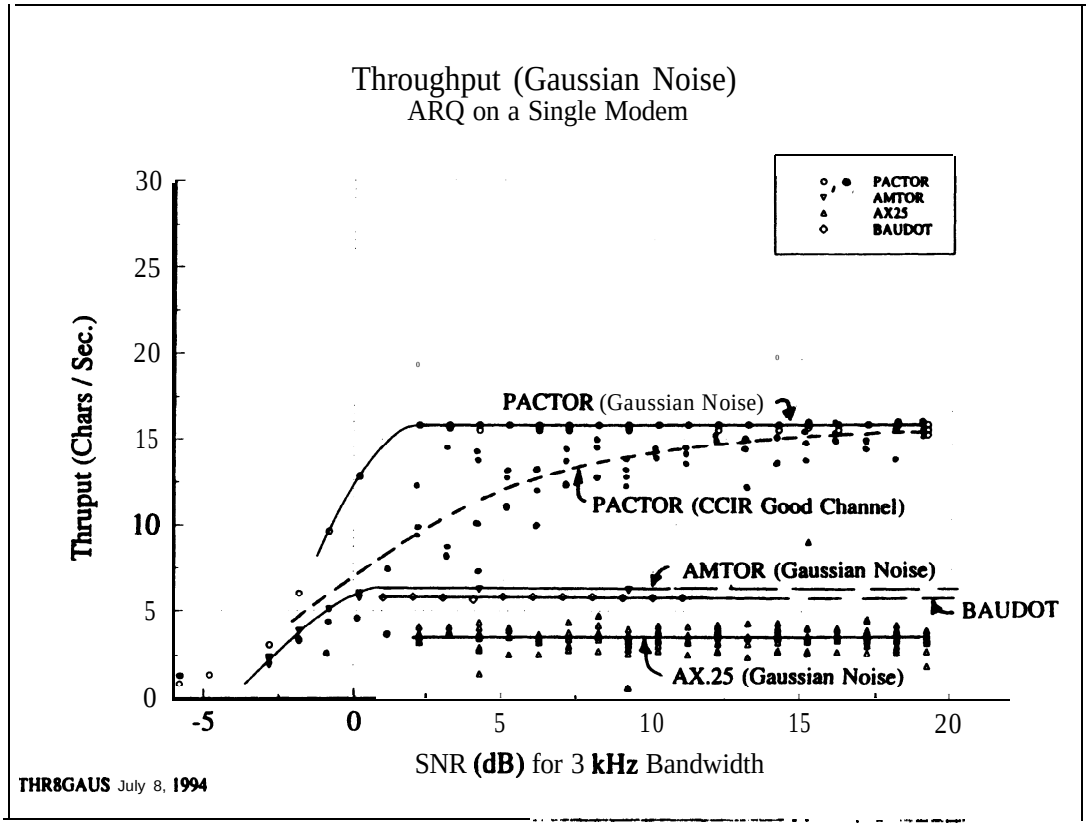
LOG (RECORDED DATA)

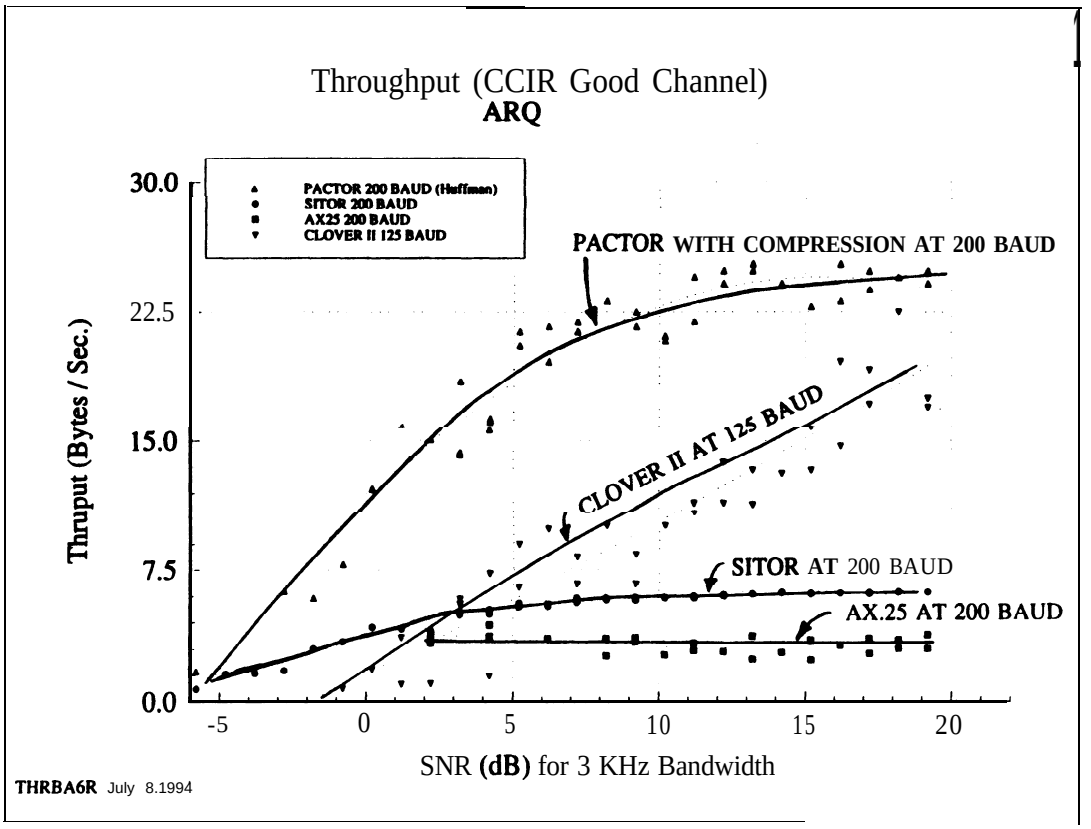
#	MODEM	DATE	START	STOP	PROCOL	MODE	FILE	CHAN	S/N	STAT
006	PKRT	07116193	15:35:33	15:40:02	AMTOR	ARQ	AM1	Q	+10	OK
007	PKRT	07/16/93	15:42:35	15:44:50	AMTOR	ARQ	AM2	Q	+10	OK
008	P K R T	07/16/93	15:53:07	16:00:12	PACTOR	F E C	AM1	Q	+05	O K
009	PKRT	07/16/93	16:00:50	16:04:21	PACTOR	FEC	AM2	Q	+05	OK
092	PKRT	07/21/93	13:52:40	13:54:57	AMTOR	ARQ	AM2	R	+20	OK
093	PKRT	07/21/93	14:02:19	14:05:50	PACTOR	FEC	AM2	R	+10	ERROR

THRUPUT (PROCESSED DATA)

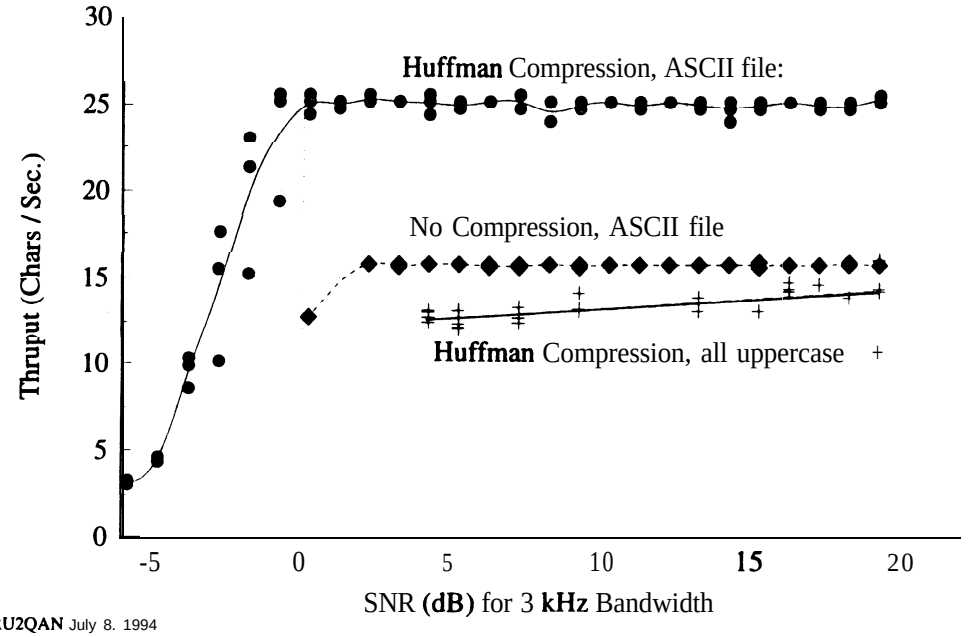
#	MODEM	DATE	START	#SEC	PROCOL	MODE	FILE	BYTES	CH	S/N	THRUPT
006	PKRT	07116193	15:35:33	269	AMTOR	ARQ	AM1	1687	Q	+10	OK 6.27
007	PKRT	07116193	15:42:35	135	AMTOR	ARQ	AM2	836	Q	+10	OK 6.19
008	PKRT	07/16/93	15:53:07	425	PACTOR	FEC	AM1	1687	Q	+05	OK 3.97
009	PKRT	07/16/93	16:00:50	211	PACTOR	FEC	AM2	836	Q	+05	OK 3.96
092	PKRT	07/21/93	13:52:40	137	AMTOR	ARQ	AM2	836	R	+20	OK 6.10
093	PKRT	07/21/93	14:02:19	211	PACTOR	FEC	AM2	836	R	+10	ERR 3.96

7/23/93


Throughput (Gaussian Noise)
ARQ on a Single Modem
PACTOR
AMTOR
AX25
BAUDOT
Thruput (Chars / Sec.)
30
25
20
15
10
5
0
PACTOR (Gaussian Noise)
PACTOR (CCIR Good Channel)
AMTOR (Gaussian Noise)
BAUDOT
AX.25 (Gaussian Noise)
-5
0
5
10
15
20
SNR (dB) for 3 kHz Bandwidth
THR8GAUS July 8, 1994

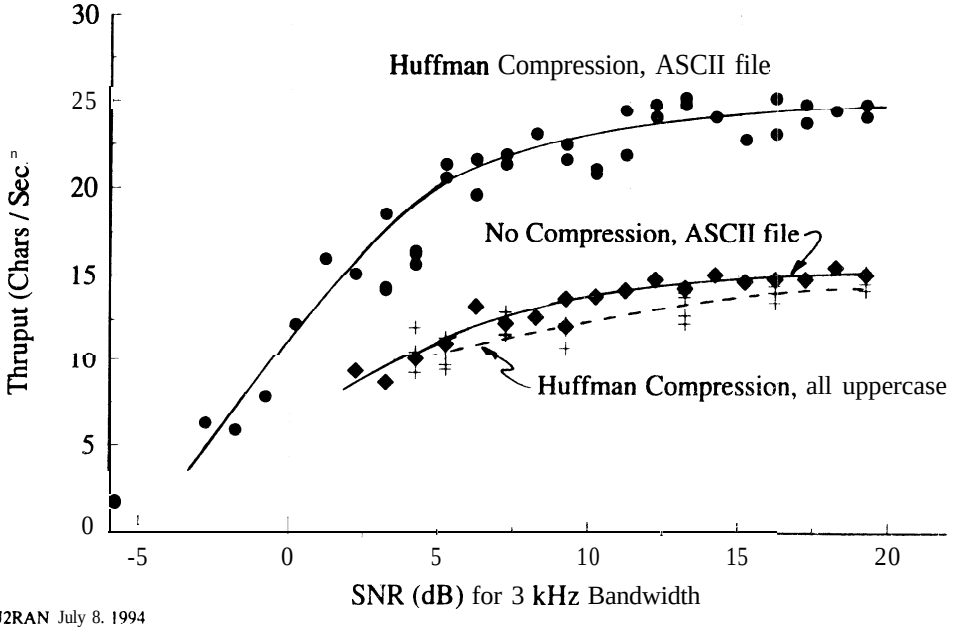

Throughput (CCIR Good Channel)
ARQ
PACTOR 200 BAUD (Huffman)
SITOR 200 BAUD
AX25 200 BAUD
CLOVER II 125 BAUD
Thruput (Bytes / Sec.)
30.0
22.5
15.0
7.5
0.0
PACTOR WITH COMPRESSION AT 200 BAUD
CLOVER II AT 125 BAUD
SITOR AT 200 BAUD
AX.25 AT 200 BAUD
-5
0
5
10
15
20
SNR (dB) for 3 KHz Bandwidth
THRBA6R July 8.1994

Throughput (Gaussian Noise)
Pactor ARQ



THRU2QAN July 8. 1994

Throughput (CCIR Good)
Pactor ARQ



THRU2RAN July 8. 1994

Notes

Notes